AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

Course Title

INTRODUCTION TO DATABASE (2108)

Semester: SPRING 2023-2024

Section: [P]

TITLE

Course Management System

Supervised By

Razuan Karim

Submitted By: Group no: D

Name	**Student_ID**
Mohammad Asad Bin Jafor	23-50088-1
Kazi Arman Alam	23-50089-1
Taki Tajuar	23-50103-1
Sk Mumitur Rahman Saba	23-50123-1

TABLE OF CONTENTS

Introduction

In a prestigious university, there’s a huge amount of student data as well as other academical and curricular activities data that represents the university’s internal integrity. These data and information is managed by different kinds of database or management systems. One of them is a Course Management System.

In this course management system, various entities collaborate seamlessly to create an efficient educational environment. Our advanced **“**Course Management System**”** handles these kinds of problems. The system manages Students, Programs, Courses, Departments, Examinations, Grades, Authority, Class Schedules, and Faculties. Each entity plays an important role, contributing to the never-ending academic activities.

Case Study

StudentID1: Name:	StudentID3: Name:	
StudentID2: Name:	StudentID4: Name:	
CO2: Understand the fundamental concepts underlying database systems and gain hands-on experience with ER diagram Case study		
PO-c2: Develop process for complex computer science and engineering problems considering cultural and societal factors.		Marks

Course Management System

A course management system that can store and manage information related to Courses, Departments, Faculties, Students, Registration, Exams, and Grades. The system would allow users to perform various operations such as creating, updating, deleting, and querying data from different tables. The system should also ensure data integrity and consistency by enforcing appropriate constraints and relationships among the tables. Course has Course_name, Course_id, More_info, and Credits. Course need to be have a registration. A course can have multiple registration, but each registration is for one course. Registration has Registration_id, Course_id, Student_id, Registration_date. Department has Department_id, Department_Name. A department can offer many courses, but each course has one department. Faculty has Faculty_id, First_name, Email. A faculty can teach multiple courses, and a course can have multiple faculties. Student has Student_id, First_name, Last_name, Email, Date_of_Birth. A student can be register in multiple courses. Exam has Exam_id, Course_id, Exam_Name, Max_Score, Exam_date**.** An exam is associated with one course, but a course can have multiple exams. Grade has Grade_id, Exam_id, Student_id, CGPA, Review. Each grade is linked with one exam but an exam can have multiple grades.

ER Diagram

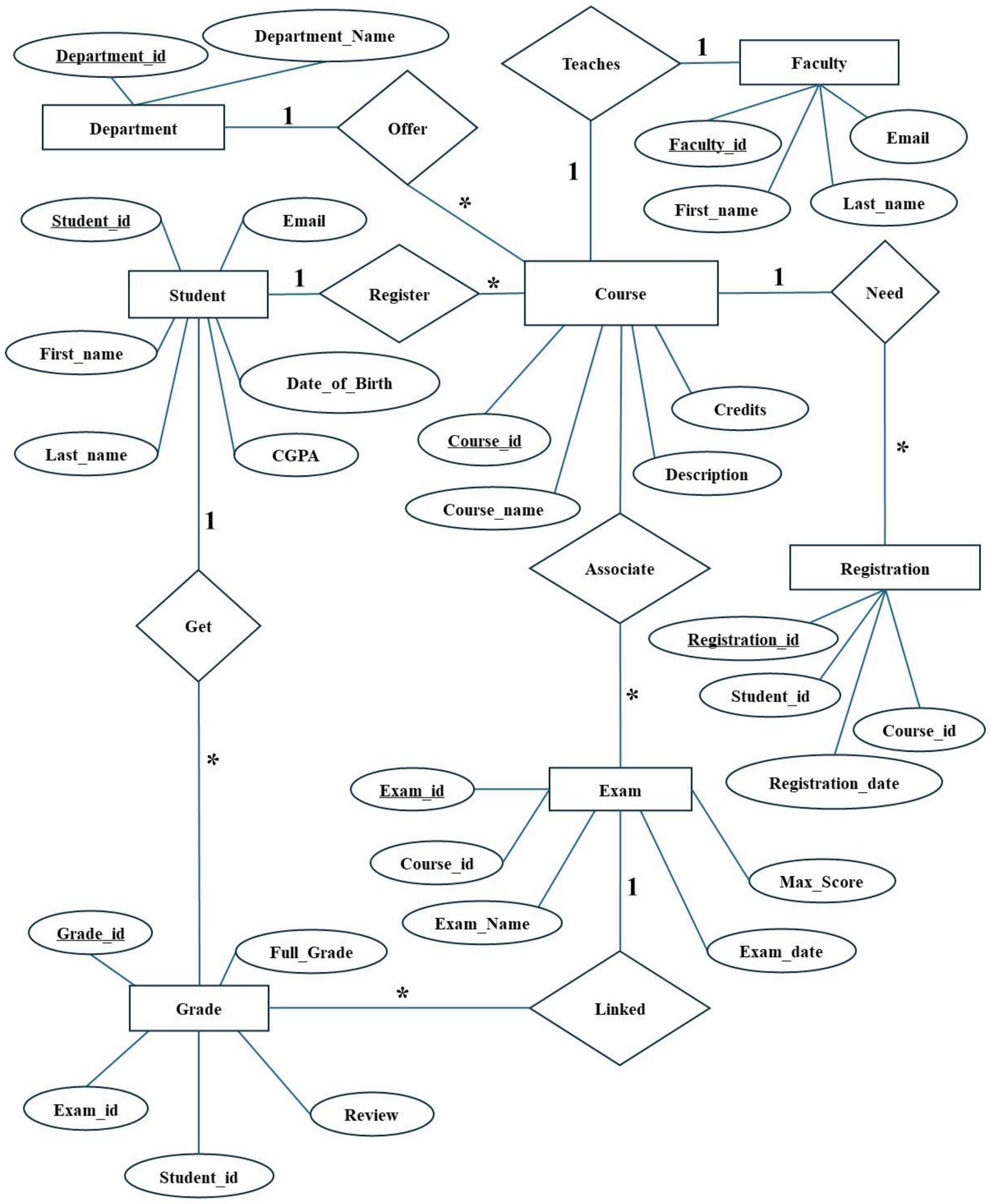


Fig 1: E-R Diagram

Normalization

1.

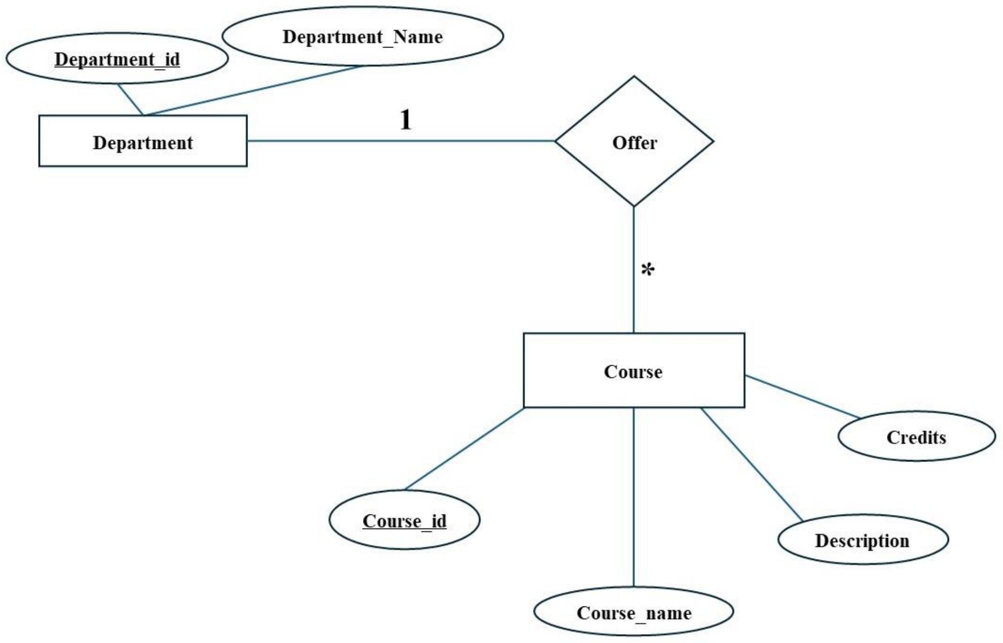


Fig 2: Department – Offer – Courses

Relation: One to Many

UNF:

1. Offer (<u>Department_ID</u>, Department Name, <u>Course_ID</u>, Course _Name, Description, Credits).

1NF:

1. <u>Course_ID</u>, Course_Name, Description, Credits, Department_ID, Department_Name.

2NF:

1. <u>Department _ID</u>, Department _Name.
2. <u>Course _ID</u>, Course _Name, Description, Credits, <u>Department _ID</u>.

3NF:

1. <u>Department _ID</u>, Department _Name.
2. <u>Course _ID</u>, Course _Name, Description, Credits, <u>Department _ID</u>.

2.

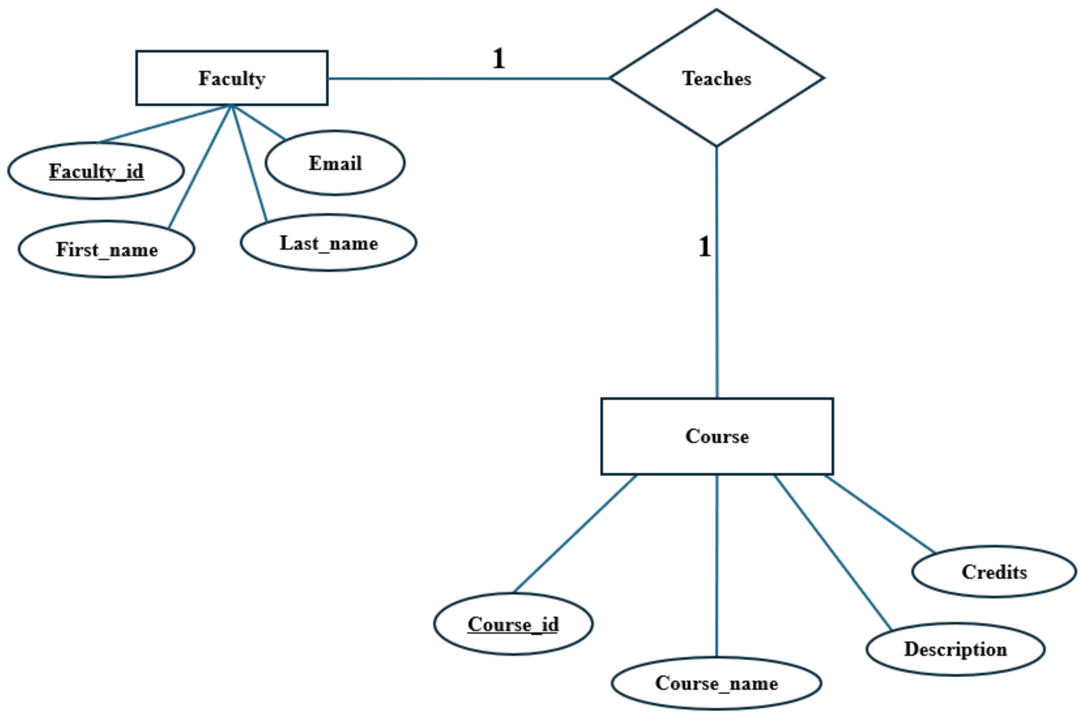


Fig 3: Faculty – Teaches – Course

Relation: One to One

UNF:

1. Teaches (Faculty _ID, First_Name, Last_Name, Email**,** Course_ID, Course_Name, Description, Credits).

1NF:

1. Faculty_ID, First _Name, Last _Name, Email**,** Course _ID, Course_Name, Description, Credits.

2NF:

1. Faculty _ID, First _Name, Last_ Name, Email.
2. Course _ID, Course _Name, Description, Credits.
3. Faculty _ID, Course _ID.

3NF:

1. Faculty _ID, First_Name, Last_Name, Email.
2. Course_ID, Course_Name, Description, Credits.
3. Faculty_ID, Course_ID.

3.

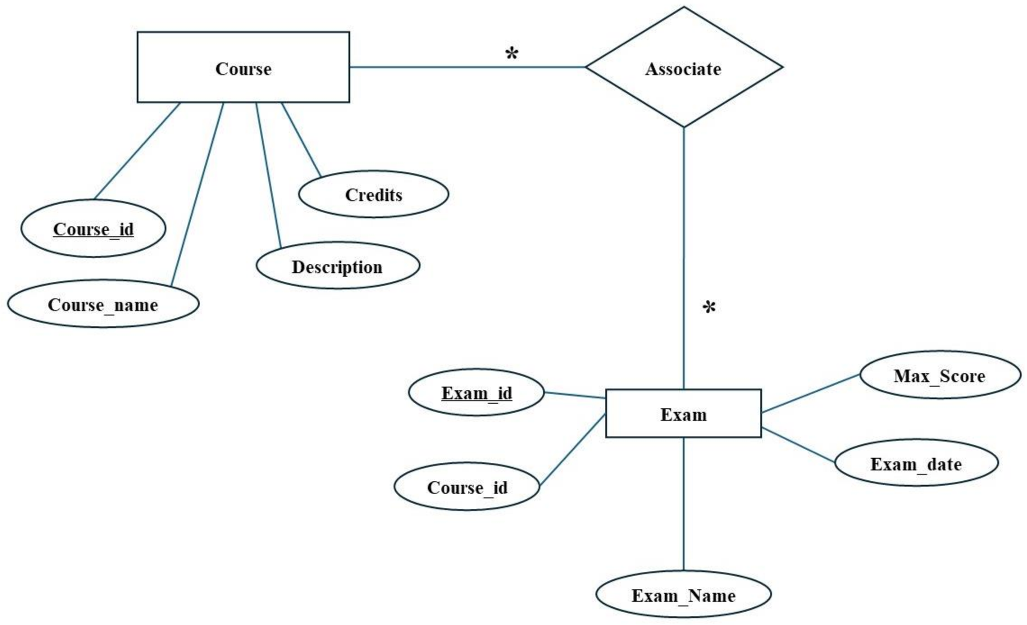


Fig 4: Course – Associate – Exam

Relation: Many to Many

UNF:

1. Associate (<u>Course_ID</u>, Course_Name, Description, Credits, <u>Exam_ID</u>, Exam_Name, Max_Score, Exam_date)

1NF:

1. <u>Course_ID</u>, Course_Name, Description, Credits, <u>Exam_ID</u>, Exam_Name, Max_Score, Exam_date

2NF:

1. <u>Course_ID</u>, Course_Name, Description, Credits
2. <u>Exam_ID</u>, Exam_Name, Max_Score, Exam_date, <u>Course_ID</u>.

3NF:

1. <u>Course_ID</u>, Course_Name, Description, Credits
2. <u>Exam_ID</u>, Exam_Name, Max_Score, Exam_date, <u>Course_ID</u>.

4.

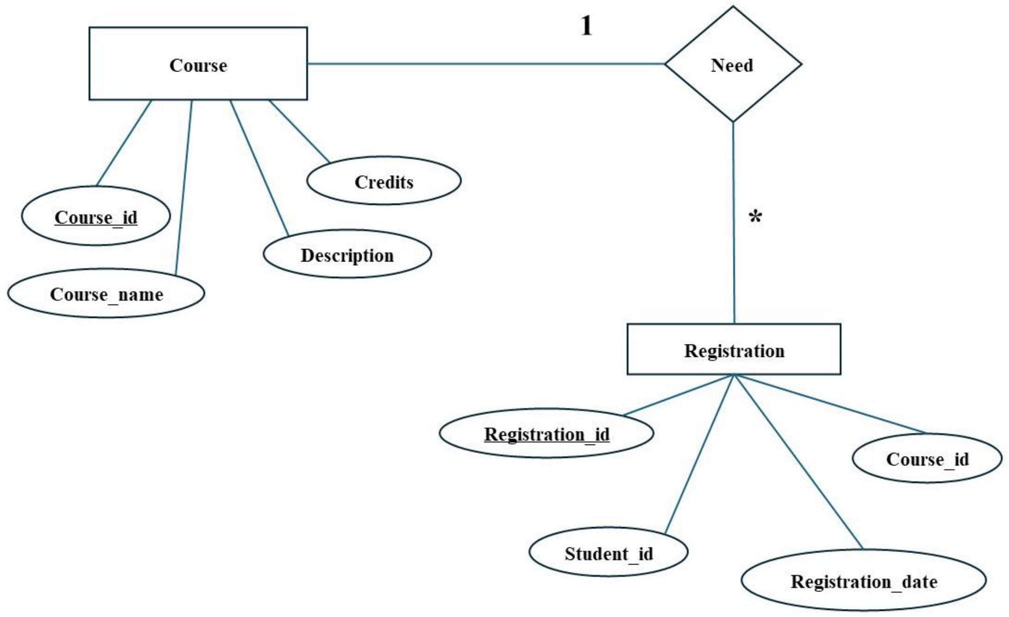


Fig 5: Course – Need – Registration

Relation: One to Many

UNF:

1. Need (<u>Course_ID</u>, Course_Name, Description, Credits, <u>Registration_ID</u>, Student_ID, Registration_Date)

1NF:

1. <u>Course_ID</u>, Course_Name, Description, Credits, <u>Registration_ID</u>, Student_ID, Registration_Date.

2NF:

1. <u>Course_ID</u>, Course_Name, Description, Credits.
2. <u>Registration_ID</u>, Course_ID, Student_ID, Registration_Date.

3NF:

1. <u>Course_ID</u>, Course_Name, Description, Credits.
2. <u>Registration_ID</u>, Course_ID, Student_ID, Registration_Date.

5.

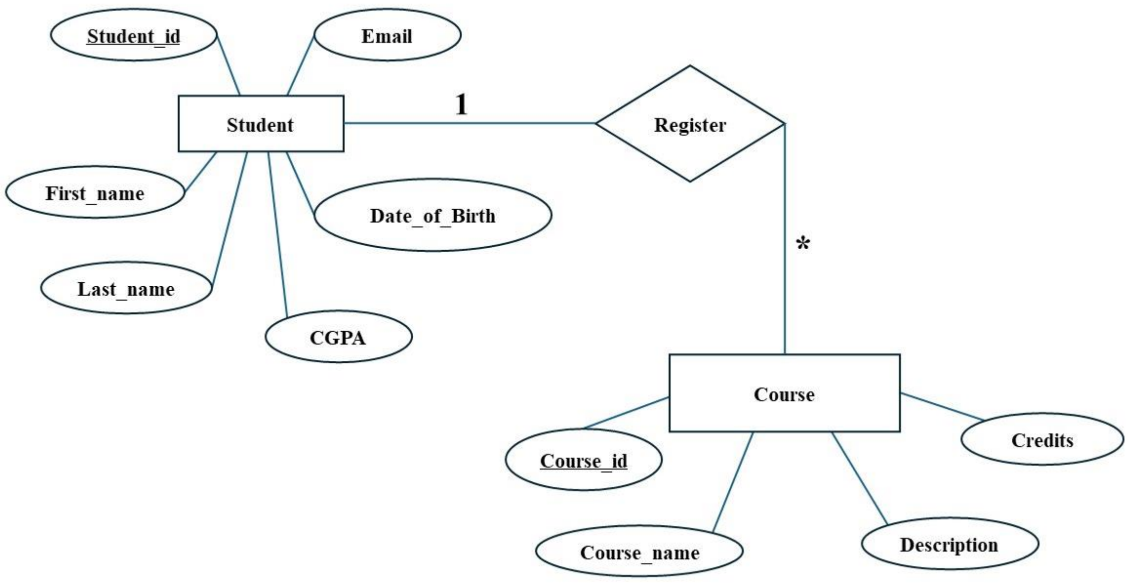


Fig 6: Student – Register – Courses

Relation: One to Many

UNF:

1. Register (CGPA, Student_ID, First_Name, Date_of_Birth, Email, Last_Name, Course_ID, Course_Name, Description, Credits)

1NF:

1. CGPA, Student_ID, First_Name, Date_of_Birth, Email, Last_Name, Course_ID, Course_Name, Description, Credits.

2NF:

1. <u>Student_ID</u>, CGPA, First_Name, Date_of_Birth, Email, Last_Name
2. <u>Course_ID</u>, Course_Name, Description, Credits, <u>Student_ID</u>.

3NF:

1. <u>Student_ID</u>, CGPA, First_Name, Date_of_Birth, Email, Last_Name
2. <u>Course_ID</u>, Course_Name, Description, Credits, <u>Student_ID</u>.

6.

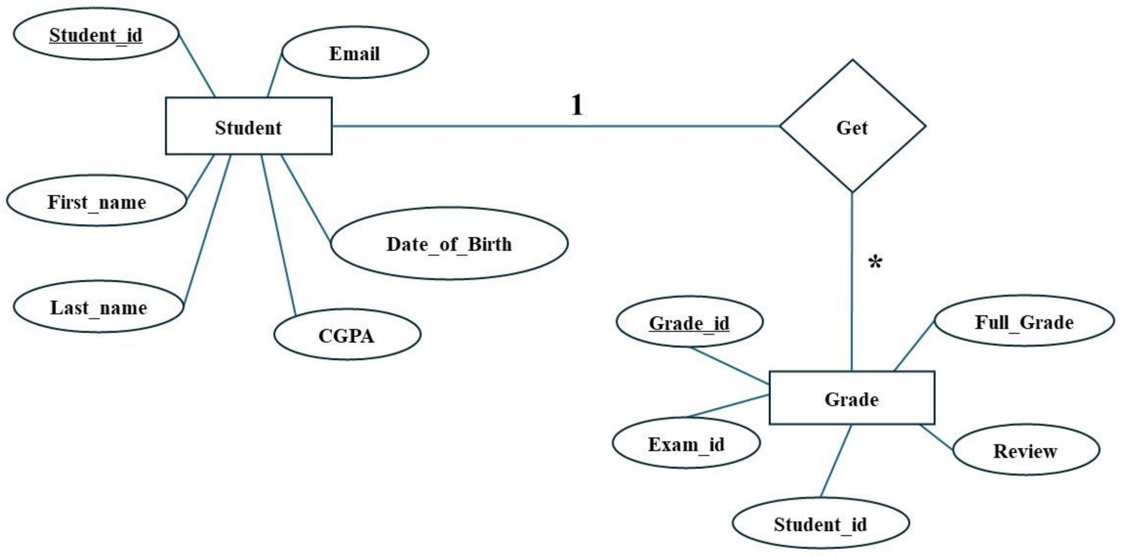


Fig 7: Student – Get – Grade

UNF:

1. Get (CGPA, Student_ID, First_Name, Date_of_Birth, Email, Grade_ID, Exam_ID, Full_Grade, Review)

1NF:

1. CGPA, Student_ID, First_Name, Date_of_Birth, Email, Grade_ID, Exam_ID, Full_Grade, Review)

2NF:

1. <u>Student_ID</u>, CGPA, First_Name, Date_of_Birth, Email, Last_Name.
2. <u>Grade_ID</u>, <u>Exam_ID</u>, <u>Student_ID</u>, Full_Grade, Review.

3NF:

1. <u>Student_ID</u>, CGPA, First_Name, Date_of_Birth, Email, Last_Name.
2. <u>Grade_ID</u>, <u>Exam_ID</u>, <u>Student_ID</u>, Full_Grade, Review.

7.

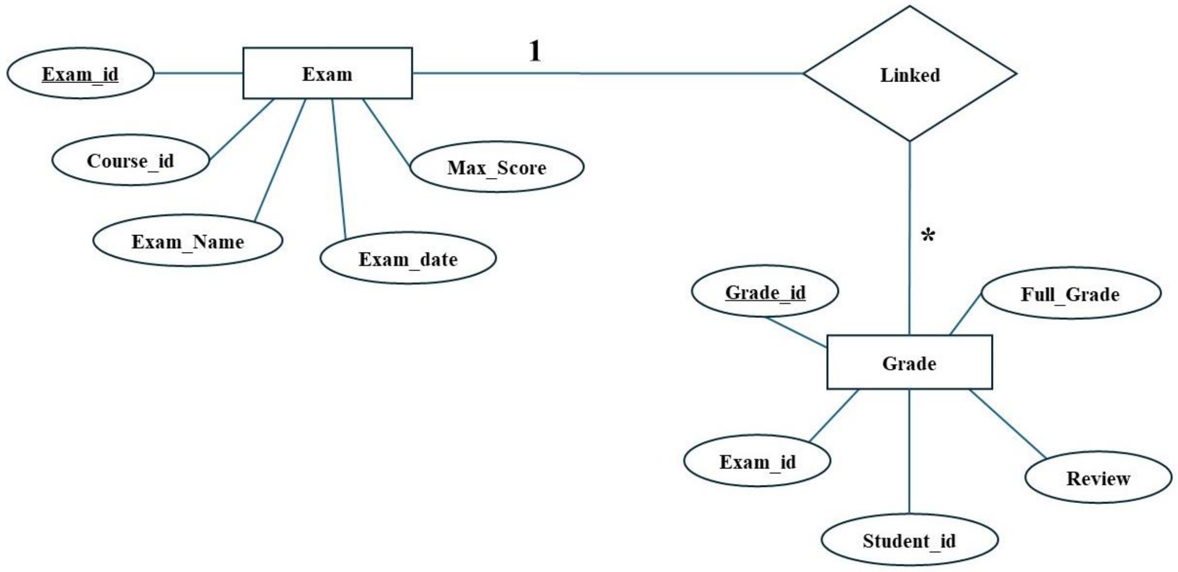


Fig 8: Exam – Linked – Grade

Relation: One to Many

UNF:

1. Linked (Exam_ID, Course_ID, Max_Score, Exam_Name, Exam_Date, Grade_ID, Student_ID, Full_Grade, Review)

1NF:

1. Exam_ID, Course_ID, Max_Score, Exam_Name, Exam_Date, Grade_ID, Student_ID, Full_Grade, Review.

2NF:

1. Exam_ID, <u>Course_ID</u>, Max_Score, Exam_Name, Exam_Date
2. <u>Grade_ID</u>, <u>Exam_ID</u>, <u>Student_ID</u>, Full_Grade, Review.

3NF:

1. Exam_ID, <u>Course_ID</u>, Max_Score, Exam_Name, Exam_Date
2. <u>Grade_ID</u>, <u>Exam_ID</u>, <u>Student_ID</u>, Full_Grade, Review.

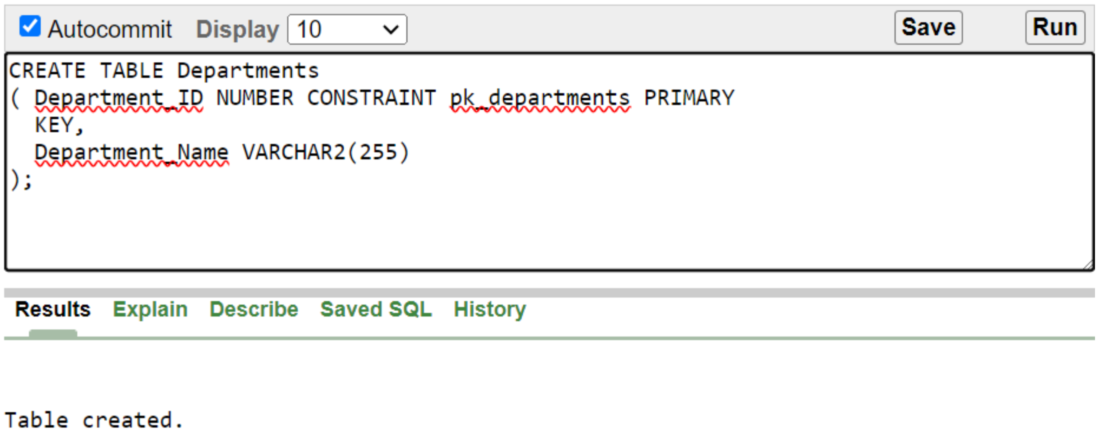
Finalization

1. Department_ID, Department_Name. (Department)
2. Course_ID, Course_Name, Description, Credits, Department_ID.(Department_Courses)
3. Faculty_ID, First_Name, Last_Name, Email (INSTRUCTOR)
4. Course_ID, Course_Name, Description, Credits. (Course)
5. Faculty_ID, Course_ID. (Faculty_Courses)
6. ~~Course_ID, Course_Name, Description, Credits. (Course)~~
7. Exam_ID, Exam_Name, Max_Score, Exam_date, Course_ID. (EXAM)
8. ~~Course_ID, Course_Name, Description, Credits~~. (COURSE)
9. Registration_ID, Course_ID, Student_ID, Registration_Date. (REGISTRATION)
10. Student_ID, CGPA, First_Name, Date_of_Birth, Email, Last_Name. (STUDENT)
11. Course_ID, Course_Name, Description, Credits, Student_ID. (Student_Courses)
12. ~~Student_ID, CGPA, First_Name, Date_of_Birth, Email, Last_Name, (STUDENT)~~
13. Grade_ID, Exam_ID, Student_ID, Full_Grade, Review. (GRADE)
14. ~~Exam_ID, Course_ID, Max_Score, Exam_Name, Exam_Date. (EXAM)~~
15. ~~Grade_ID, Exam_ID, Student_ID, Full_Grade, Review. (GRADE)~~

Table Creation (DDL Operations)

StudentID1: Name:	StudentID3: Name:	
StudentID2: Name:	StudentID4: Name:	
CO4: Creating DML, DDL using Oracle and connection with ODBC/JDBC for existing JAVA application		
PO-e-2: Use modern engineering and IT tools for prediction and modeling of complex computer science and engineering problem		Marks

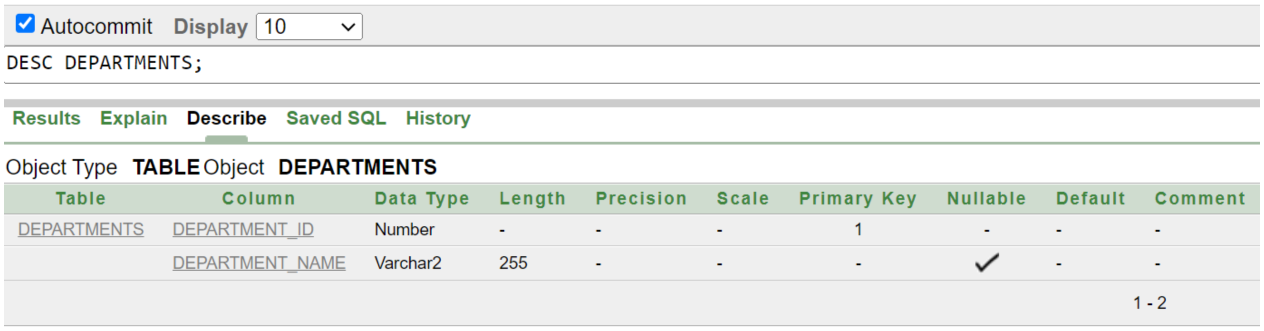
Table: Departments



```
CREATE TABLE Departments
( Department_ID NUMBER CONSTRAINT pk_departments PRIMARY
  KEY,
  Department_Name VARCHAR2(255)
);
```

```
Table created.
```

Fig 9: Command for Departments table



```
DESC DEPARTMENTS;
```

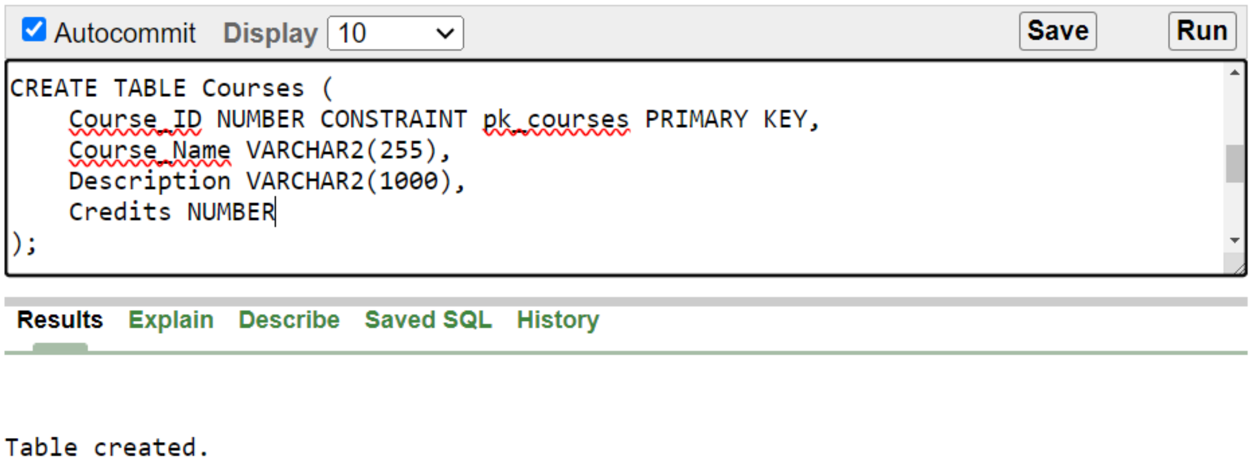
Object Type **TABLE** Object **DEPARTMENTS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENTS	DEPARTMENT_ID	Number	-	-	-	1	-	-	-
	DEPARTMENT_NAME	Varchar2	255	-	-	-	✓	-	-

1 - 2

Fig 10: Departments table

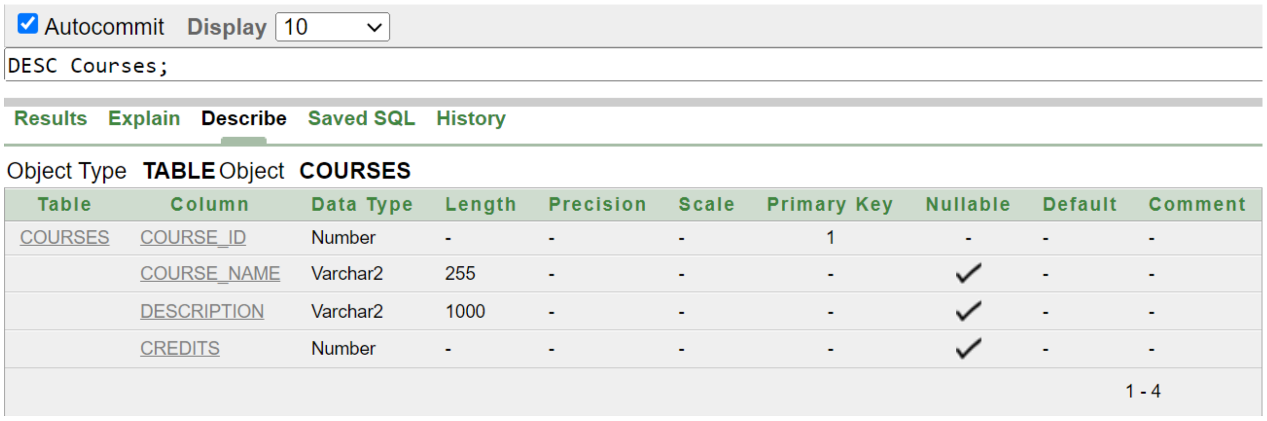
Table: Courses



```
CREATE TABLE Courses (
    Course_ID NUMBER CONSTRAINT pk_courses PRIMARY KEY,
    Course_Name VARCHAR2(255),
    Description VARCHAR2(1000),
    Credits NUMBER
);
```

```
Table created.
```

Fig 11: Command for Courses table



```
DESC Courses;
```

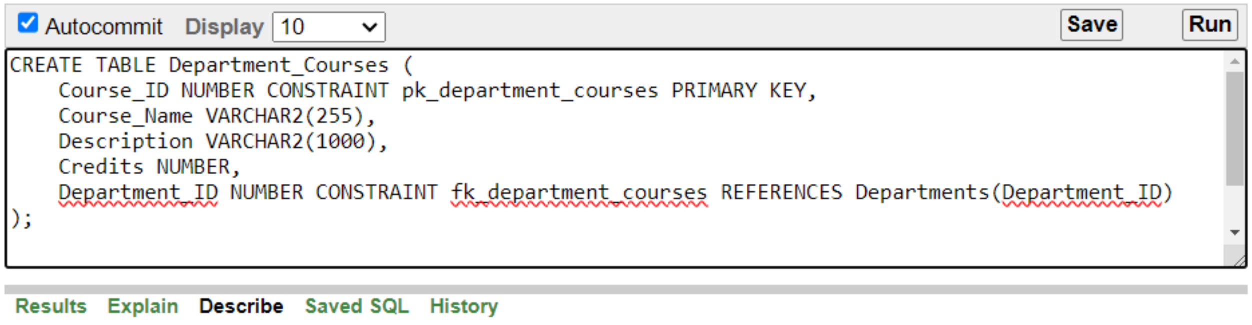
Object Type **TABLE** Object **COURSES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COURSES	COURSE_ID	Number	-	-	-	1	-	-	-
	COURSE_NAME	Varchar2	255	-	-	-	✓	-	-
	DESCRIPTION	Varchar2	1000	-	-	-	✓	-	-
	CREDITS	Number	-	-	-	-	✓	-	-

1 - 4

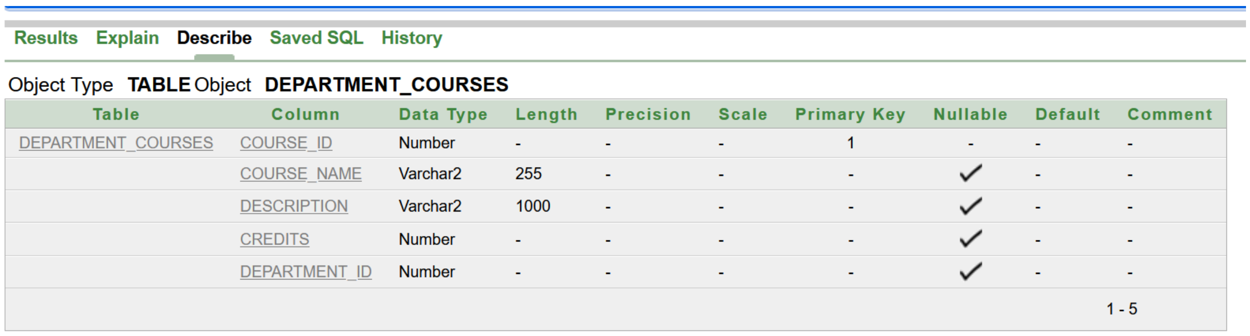
Fig 12: Courses table

Table: Department_Courses (Association Table)



```
CREATE TABLE Department_Courses (
    Course_ID NUMBER CONSTRAINT pk_department_courses PRIMARY KEY,
    Course_Name VARCHAR2(255),
    Description VARCHAR2(1000),
    Credits NUMBER,
    Department_ID NUMBER CONSTRAINT fk_department_courses REFERENCES Departments(Department_ID)
);
```

Fig 13: Command for Department_Courses table



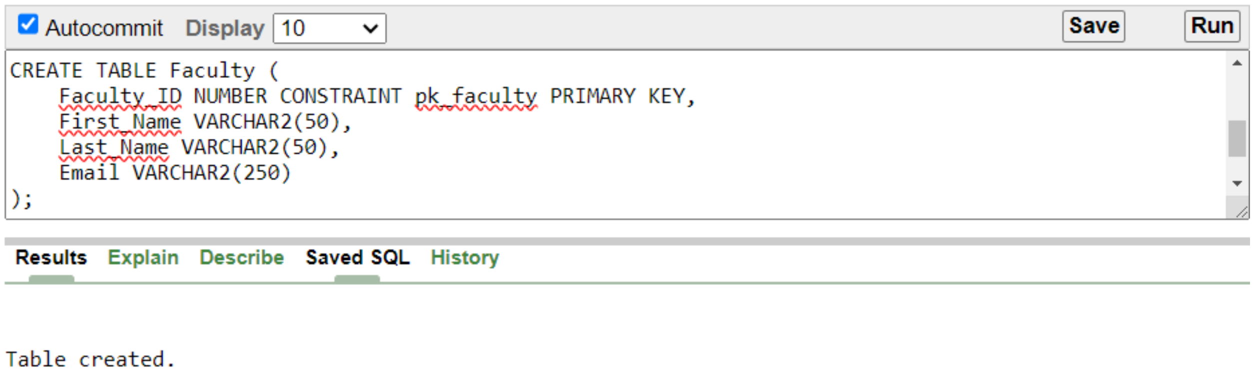
Object Type **TABLE** Object **DEPARTMENT_COURSES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT_COURSES	COURSE_ID	Number	-	-	-	1	-	-	-
	COURSE_NAME	Varchar2	255	-	-	-	✓	-	-
	DESCRIPTION	Varchar2	1000	-	-	-	✓	-	-
	CREDITS	Number	-	-	-	-	✓	-	-
	DEPARTMENT_ID	Number	-	-	-	-	✓	-	-

1 - 5

Fig 14: Department_courses table

Table: Faculty



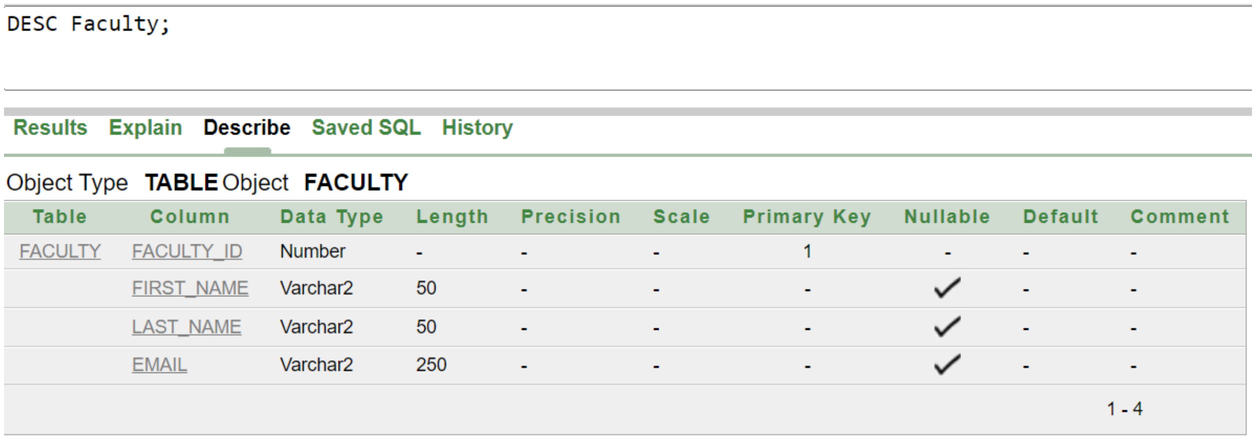
Autocommit Display 10 Save Run

```
CREATE TABLE Faculty (
    Faculty_ID NUMBER CONSTRAINT pk_faculty PRIMARY KEY,
    First_Name VARCHAR2(50),
    Last_Name VARCHAR2(50),
    Email VARCHAR2(250)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

Fig 15: Command for Faculty table



```
DESC Faculty;
```

Results Explain Describe Saved SQL History

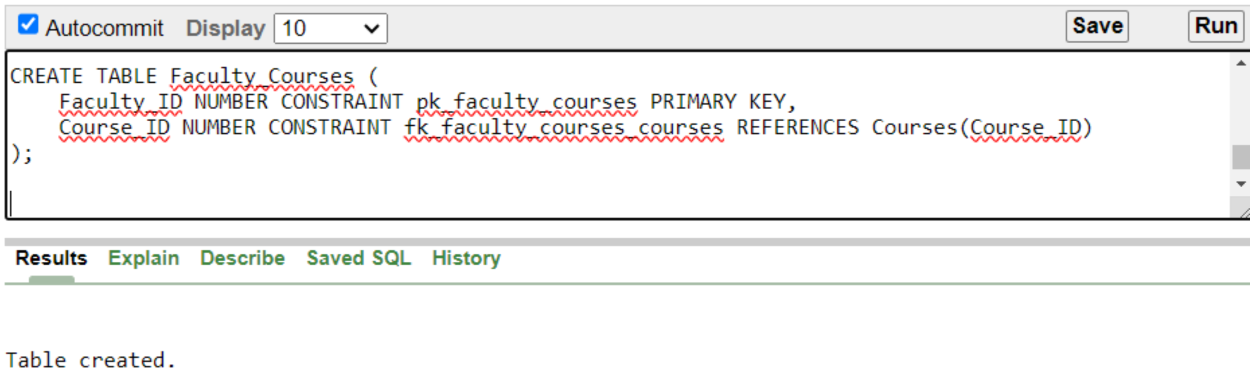
Object Type **TABLE** Object **FACULTY**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FACULTY	FACULTY_ID	Number	-	-	-	1	-	-	-
	FIRST_NAME	Varchar2	50	-	-	-	✓	-	-
	LAST_NAME	Varchar2	50	-	-	-	✓	-	-
	EMAIL	Varchar2	250	-	-	-	✓	-	-

1 - 4

Fig 16: Faculty table

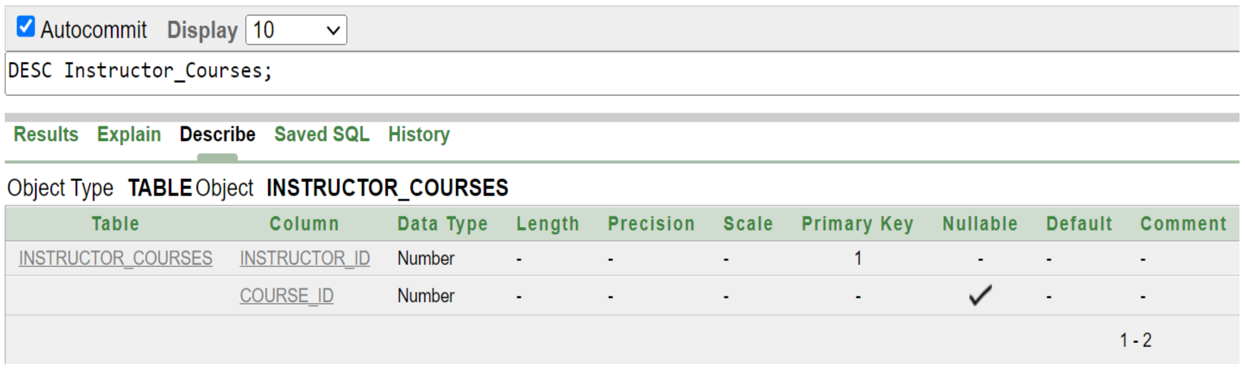
Table: Faculty_Courses (Association Table)



```
CREATE TABLE Faculty_Courses (
    Faculty_ID NUMBER CONSTRAINT pk_faculty_courses PRIMARY KEY,
    Course_ID NUMBER CONSTRAINT fk_faculty_courses_courses REFERENCES Courses(Course_ID)
);
```

```
Table created.
```

Fig 17: Command for Faculty Courses table



```
DESC Instructor_Courses;
```

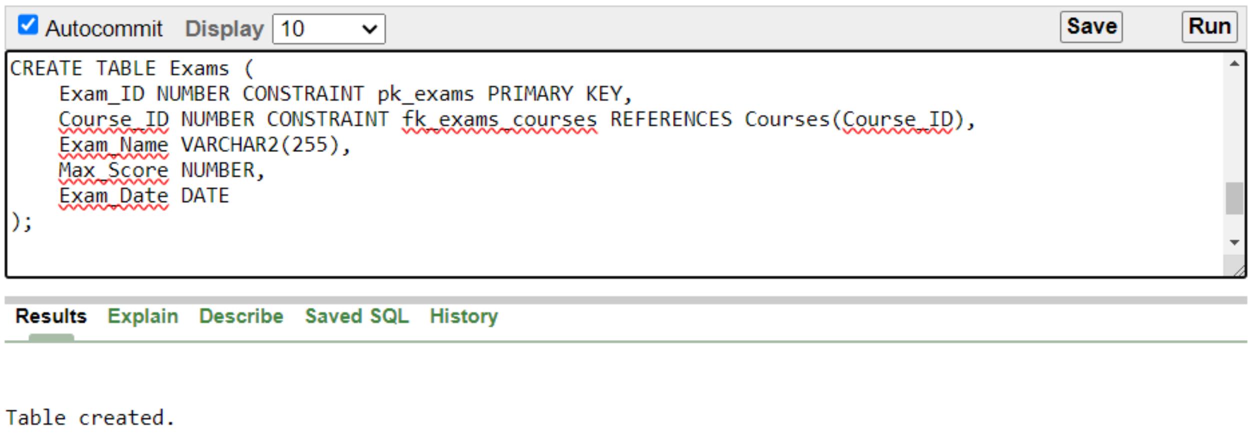
Object Type **TABLE** Object **INSTRUCTOR_COURSES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
INSTRUCTOR_COURSES	INSTRUCTOR_ID	Number	-	-	-	1	-	-	-
	COURSE_ID	Number	-	-	-	-	✓	-	-

1 - 2

Fig 18: Faculty_Courses table

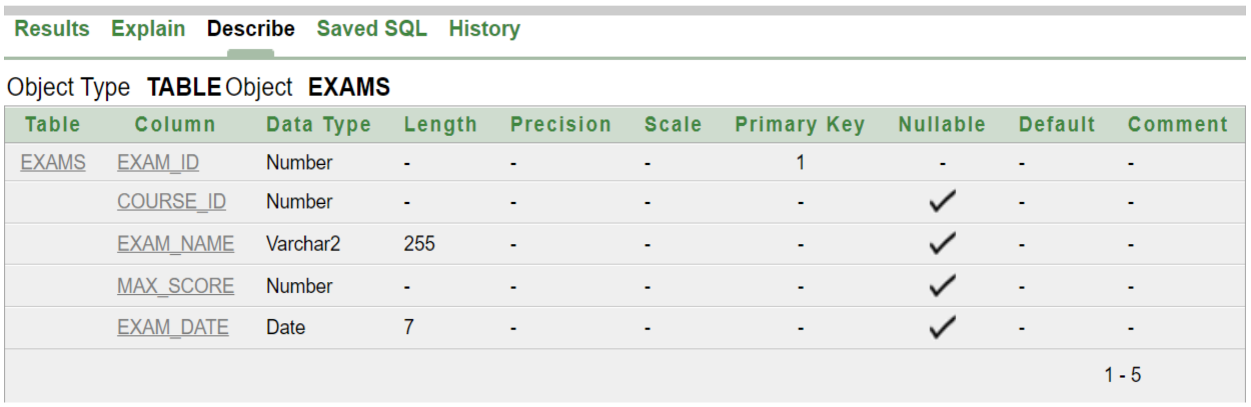
Table: Exams Table:



```
CREATE TABLE Exams (
    Exam_ID NUMBER CONSTRAINT pk_exams PRIMARY KEY,
    Course_ID NUMBER CONSTRAINT fk_exams_courses REFERENCES Courses(Course_ID),
    Exam_Name VARCHAR2(255),
    Max_Score NUMBER,
    Exam_Date DATE
);
```

```
Table created.
```

Fig 19: Command for Exams table



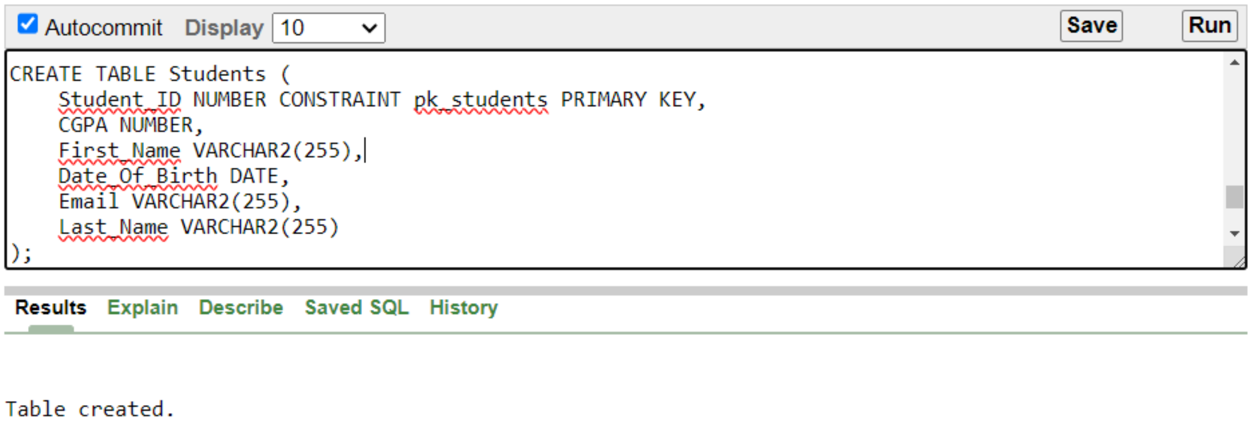
Object Type **TABLE** Object **EXAMS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EXAMS	EXAM_ID	Number	-	-	-	1	-	-	-
	COURSE_ID	Number	-	-	-	-	✓	-	-
	EXAM_NAME	Varchar2	255	-	-	-	✓	-	-
	MAX_SCORE	Number	-	-	-	-	✓	-	-
	EXAM_DATE	Date	7	-	-	-	✓	-	-

1 - 5

Fig 20: Exams table

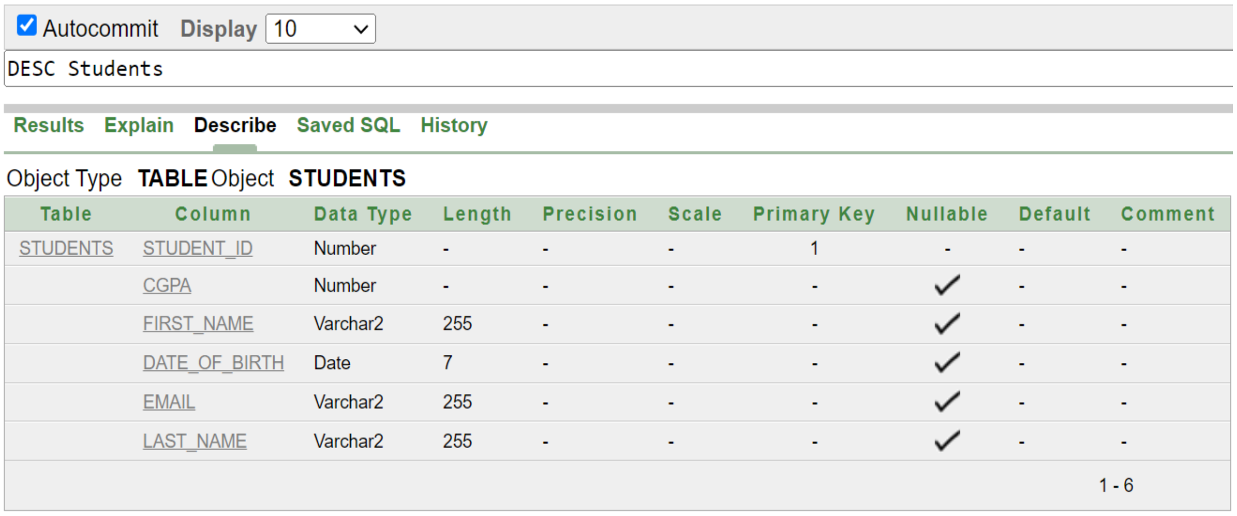
Table: Students



```
CREATE TABLE Students (
    Student_ID NUMBER CONSTRAINT pk_students PRIMARY KEY,
    CGPA NUMBER,
    First_Name VARCHAR2(255),
    Date_Of_Birth DATE,
    Email VARCHAR2(255),
    Last_Name VARCHAR2(255)
);
```

```
Table created.
```

Fig 21: Command for Students table



```
DESC Students
```

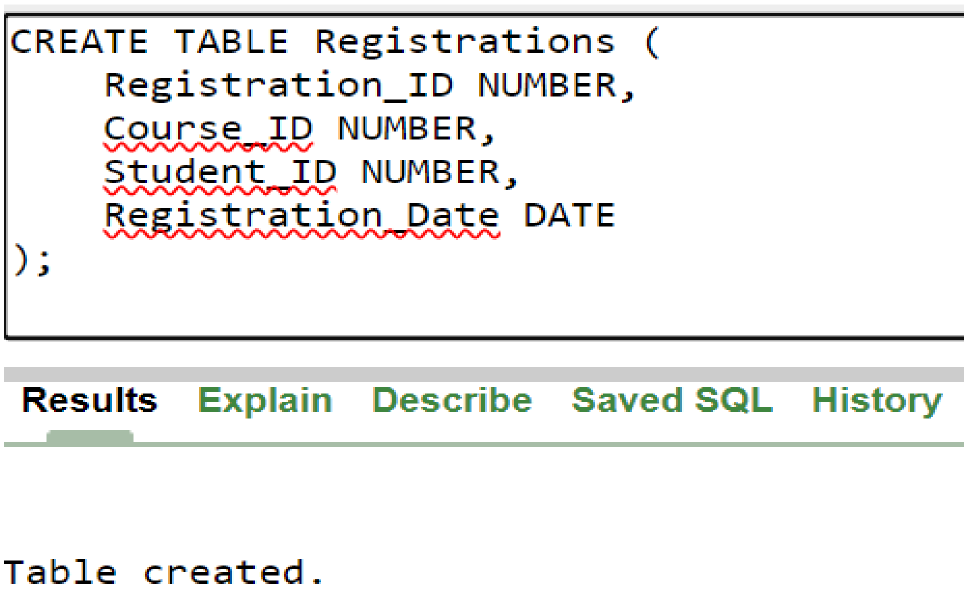
Object Type **TABLE** Object **STUDENTS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENTS	STUDENT_ID	Number	-	-	-	1	-	-	-
	CGPA	Number	-	-	-	-	✓	-	-
	FIRST_NAME	Varchar2	255	-	-	-	✓	-	-
	DATE_OF_BIRTH	Date	7	-	-	-	✓	-	-
	EMAIL	Varchar2	255	-	-	-	✓	-	-
	LAST_NAME	Varchar2	255	-	-	-	✓	-	-

1 - 6

Fig 22: Students table

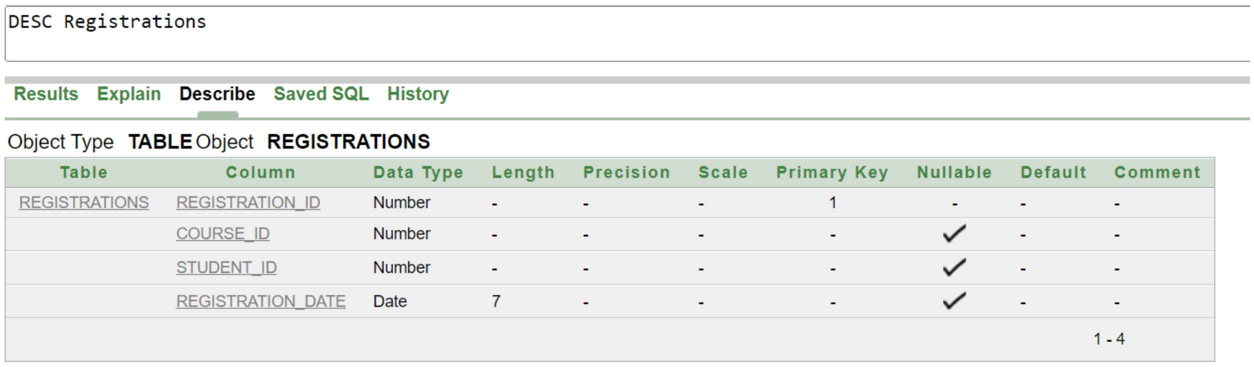
Table: Registrations



```
CREATE TABLE Registrations (
    Registration_ID NUMBER,
    Course_ID NUMBER,
    Student_ID NUMBER,
    Registration_Date DATE
);
```

```
Table created.
```

Fig 23: Command for Registration table



```
DESC Registrations
```

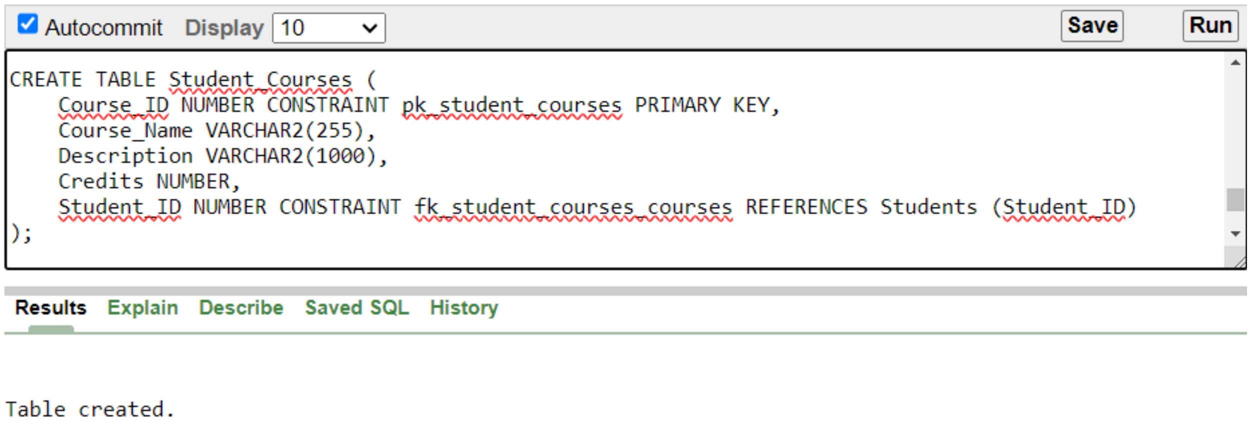
Object Type **TABLE** Object **REGISTRATIONS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REGISTRATIONS	REGISTRATION_ID	Number	-	-	-	1	-	-	-
	COURSE_ID	Number	-	-	-	-	✓	-	-
	STUDENT_ID	Number	-	-	-	-	✓	-	-
	REGISTRATION_DATE	Date	7	-	-	-	✓	-	-

1 - 4

Fig 24: Registration table

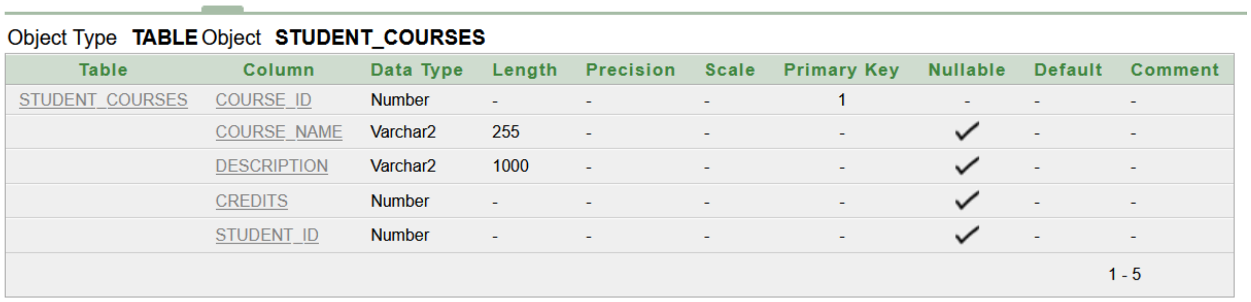
Table: Student_Courses (Association Table)



```
CREATE TABLE Student_Courses (
    Course_ID NUMBER CONSTRAINT pk_student_courses PRIMARY KEY,
    Course_Name VARCHAR2(255),
    Description VARCHAR2(1000),
    Credits NUMBER,
    Student_ID NUMBER CONSTRAINT fk_student_courses_courses REFERENCES Students (Student_ID)
);
```

```
Table created.
```

Fig 25: Command for Student_Courses table



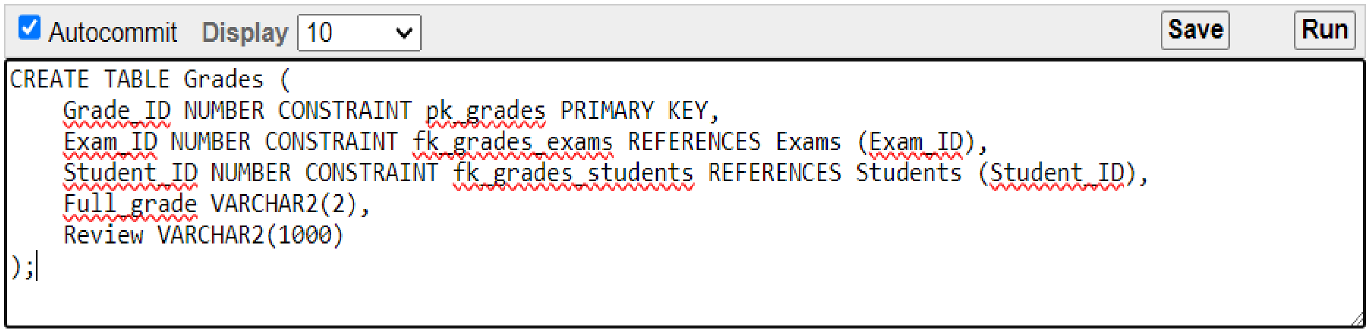
Object Type **TABLE** Object **STUDENT_COURSES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT_COURSES	COURSE_ID	Number	-	-	-	1	-	-	-
	COURSE_NAME	Varchar2	255	-	-	-	✓	-	-
	DESCRIPTION	Varchar2	1000	-	-	-	✓	-	-
	CREDITS	Number	-	-	-	-	✓	-	-
	STUDENT_ID	Number	-	-	-	-	✓	-	-

1 - 5

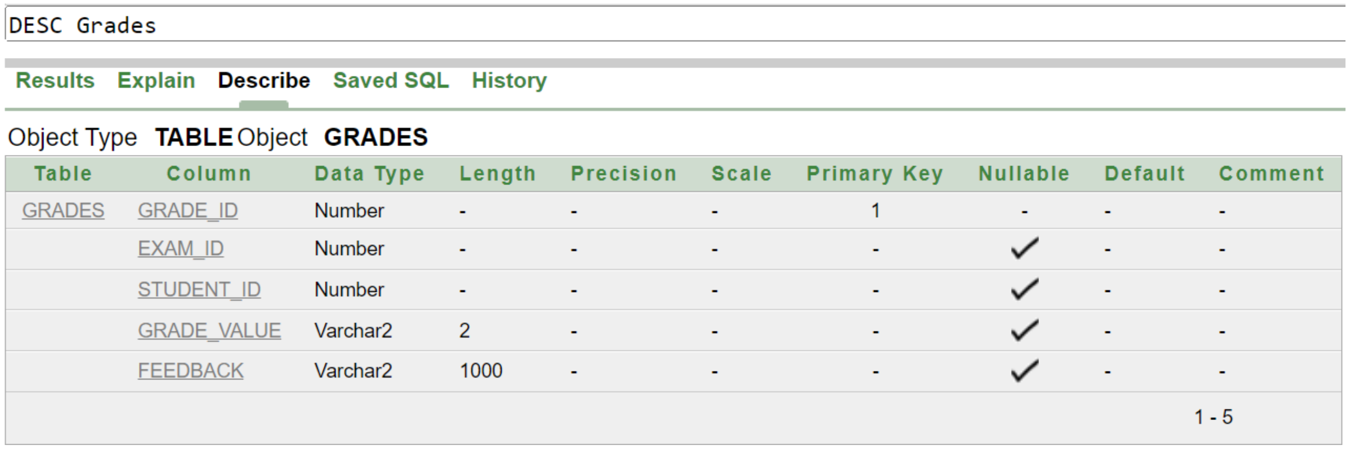
Fig 25: Student_Courses table

Table: Grades



```
CREATE TABLE Grades (
    Grade_ID NUMBER CONSTRAINT pk_grades PRIMARY KEY,
    Exam_ID NUMBER CONSTRAINT fk_grades_exams REFERENCES Exams (Exam_ID),
    Student_ID NUMBER CONSTRAINT fk_grades_students REFERENCES Students (Student_ID),
    Full_grade VARCHAR2(2),
    Review VARCHAR2(1000)
);
```

***Fig 27:* Command for Grades table**



```
DESC Grades
```

Object Type **TABLE** Object **GRADES**

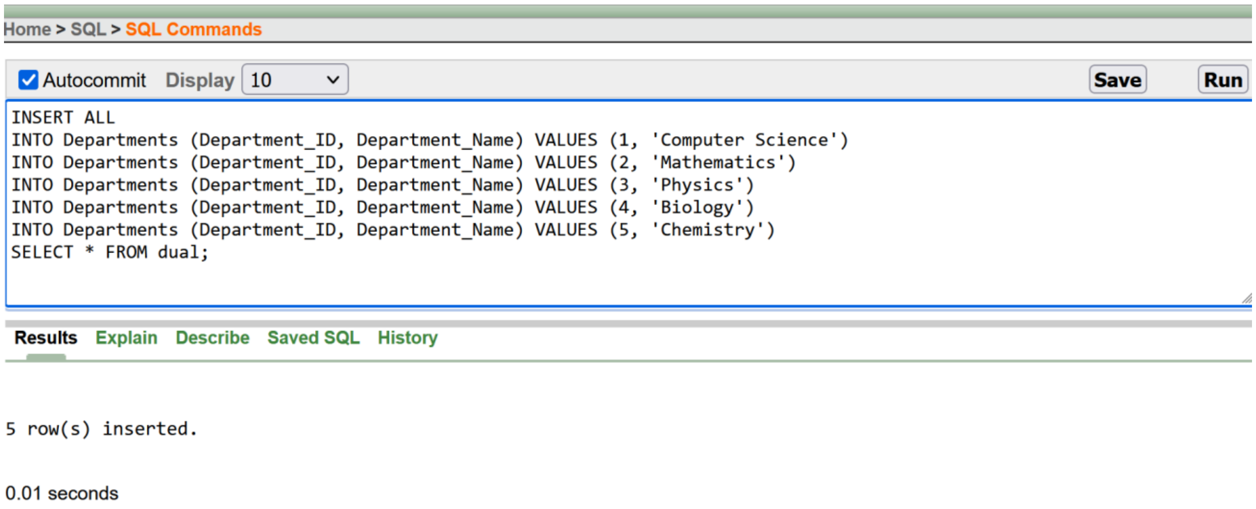
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
GRADES	GRADE_ID	Number	-	-	-	1	-	-	-
	EXAM_ID	Number	-	-	-	-	✓	-	-
	STUDENT_ID	Number	-	-	-	-	✓	-	-
	GRADE_VALUE	Varchar2	2	-	-	-	✓	-	-
	FEEDBACK	Varchar2	1000	-	-	-	✓	-	-

1 - 5

Fig 28: Grades table

VALUE INSERTION

Value insertion of Table: Departments:

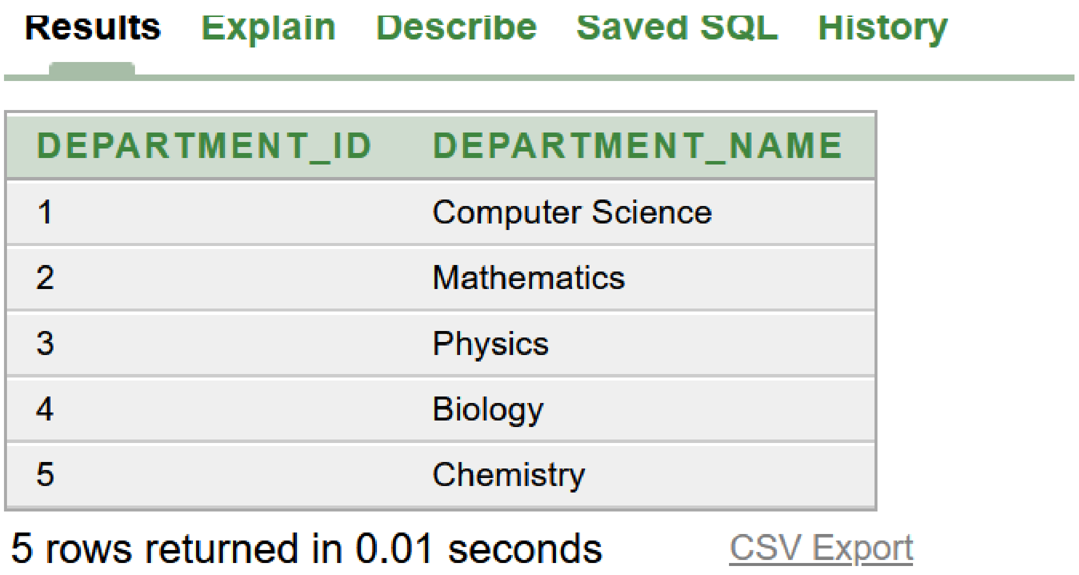


```
INSERT ALL
INTO Departments (Department_ID, Department_Name) VALUES (1, 'Computer Science')
INTO Departments (Department_ID, Department_Name) VALUES (2, 'Mathematics')
INTO Departments (Department_ID, Department_Name) VALUES (3, 'Physics')
INTO Departments (Department_ID, Department_Name) VALUES (4, 'Biology')
INTO Departments (Department_ID, Department_Name) VALUES (5, 'Chemistry')
SELECT * FROM dual;
```

```
5 row(s) inserted.

0.01 seconds
```

Fig 29: Value insertion command for table Departments.

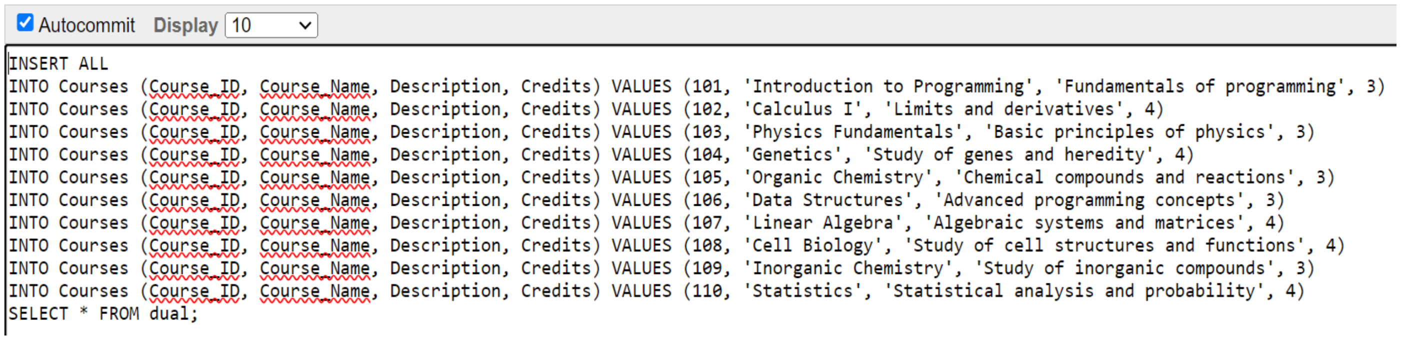


DEPARTMENT_ID	DEPARTMENT_NAME
1	Computer Science
2	Mathematics
3	Physics
4	Biology
5	Chemistry

5 rows returned in 0.01 seconds CSV Export

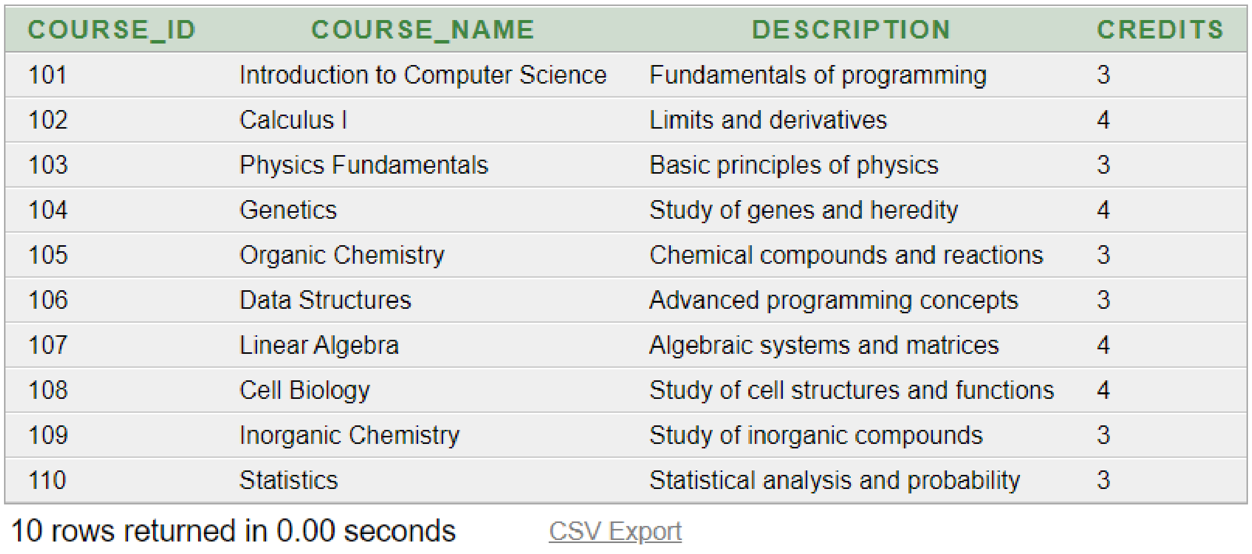
Fig30: All data of table Departments.

Value insertion of Table: Courses



```
INSERT ALL
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (101, 'Introduction to Programming', 'Fundamentals of programming', 3)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (102, 'Calculus I', 'Limits and derivatives', 4)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (103, 'Physics Fundamentals', 'Basic principles of physics', 3)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (104, 'Genetics', 'Study of genes and heredity', 4)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (105, 'Organic Chemistry', 'Chemical compounds and reactions', 3)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (106, 'Data Structures', 'Advanced programming concepts', 3)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (107, 'Linear Algebra', 'Algebraic systems and matrices', 4)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (108, 'Cell Biology', 'Study of cell structures and functions', 4)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (109, 'Inorganic Chemistry', 'Study of inorganic compounds', 3)
INTO Courses (Course_ID, Course_Name, Description, Credits) VALUES (110, 'Statistics', 'Statistical analysis and probability', 4)
SELECT * FROM dual;
```

Fig 31: Value insertion command for table Courses.

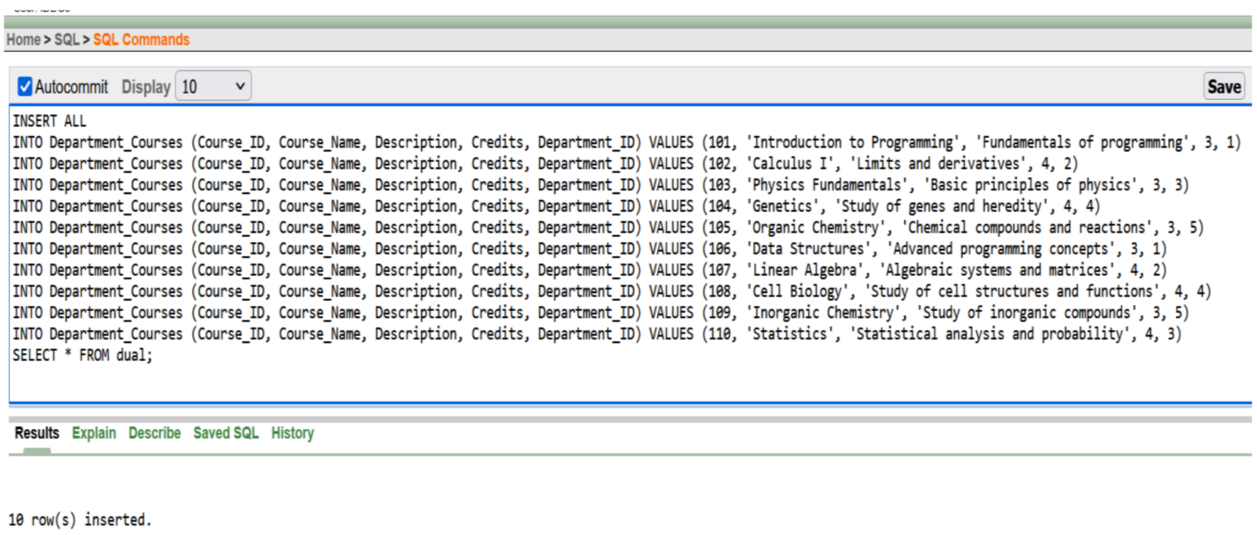


COURSE_ID	COURSE_NAME	DESCRIPTION	CREDITS
101	Introduction to Computer Science	Fundamentals of programming	3
102	Calculus I	Limits and derivatives	4
103	Physics Fundamentals	Basic principles of physics	3
104	Genetics	Study of genes and heredity	4
105	Organic Chemistry	Chemical compounds and reactions	3
106	Data Structures	Advanced programming concepts	3
107	Linear Algebra	Algebraic systems and matrices	4
108	Cell Biology	Study of cell structures and functions	4
109	Inorganic Chemistry	Study of inorganic compounds	3
110	Statistics	Statistical analysis and probability	3

10 rows returned in 0.00 seconds

Fig 32: All data of table Courses.

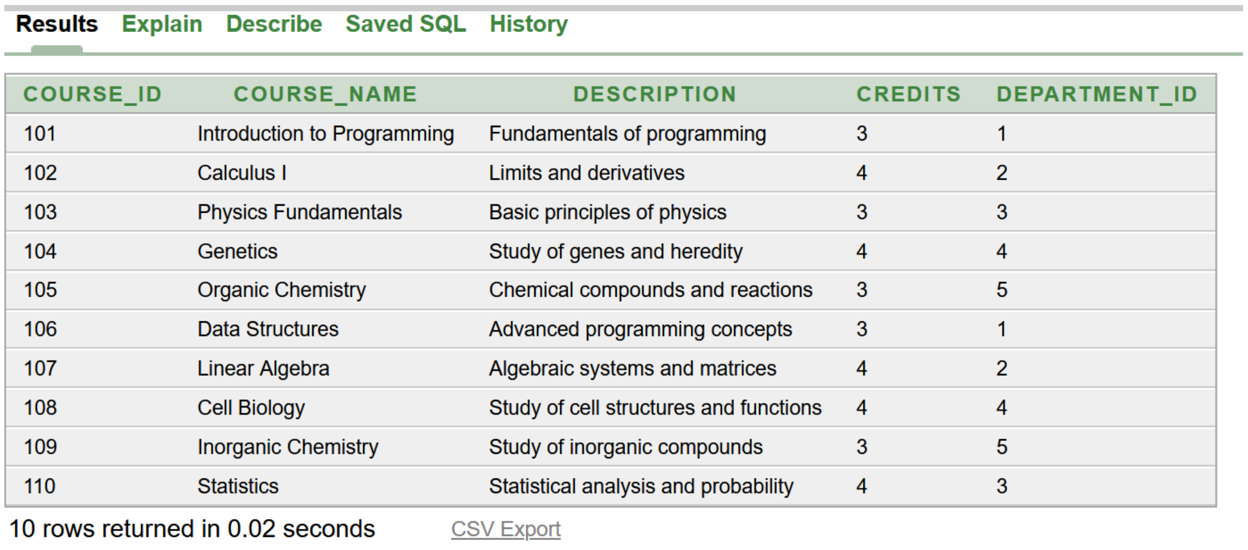
Value insertion of Table: Department_Courses



```
INSERT ALL
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (101, 'Introduction to Programming', 'Fundamentals of programming', 3, 1)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (102, 'Calculus I', 'Limits and derivatives', 4, 2)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (103, 'Physics Fundamentals', 'Basic principles of physics', 3, 3)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (104, 'Genetics', 'Study of genes and heredity', 4, 4)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (105, 'Organic Chemistry', 'Chemical compounds and reactions', 3, 5)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (106, 'Data Structures', 'Advanced programming concepts', 3, 1)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (107, 'Linear Algebra', 'Algebraic systems and matrices', 4, 2)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (108, 'Cell Biology', 'Study of cell structures and functions', 4, 4)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (109, 'Inorganic Chemistry', 'Study of inorganic compounds', 3, 5)
INTO Department_Courses (Course_ID, Course_Name, Description, Credits, Department_ID) VALUES (110, 'Statistics', 'Statistical analysis and probability', 4, 3)
SELECT * FROM dual;
```

```
10 row(s) inserted.
```

Fig 33: Value insertion command for table Departments_Courses.

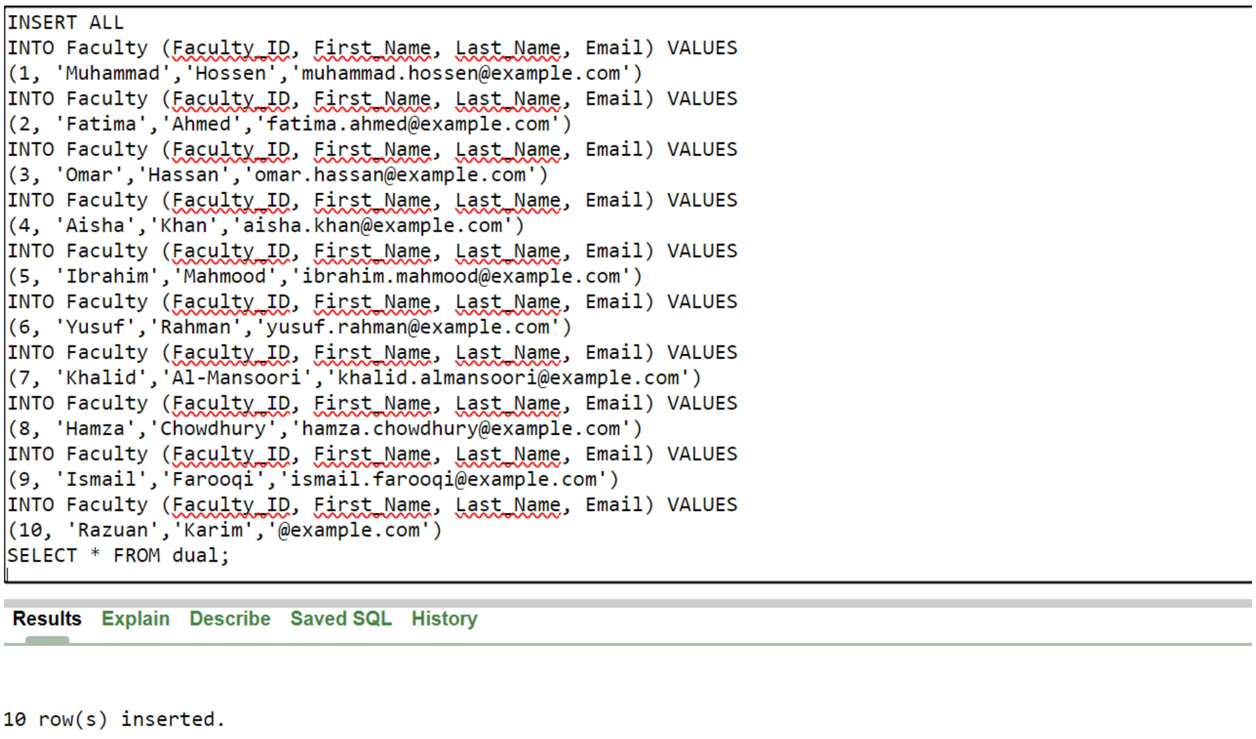


COURSE_ID	COURSE_NAME	DESCRIPTION	CREDITS	DEPARTMENT_ID
101	Introduction to Programming	Fundamentals of programming	3	1
102	Calculus I	Limits and derivatives	4	2
103	Physics Fundamentals	Basic principles of physics	3	3
104	Genetics	Study of genes and heredity	4	4
105	Organic Chemistry	Chemical compounds and reactions	3	5
106	Data Structures	Advanced programming concepts	3	1
107	Linear Algebra	Algebraic systems and matrices	4	2
108	Cell Biology	Study of cell structures and functions	4	4
109	Inorganic Chemistry	Study of inorganic compounds	3	5
110	Statistics	Statistical analysis and probability	4	3

10 rows returned in 0.02 seconds CSV Export

Fig 34: All data of table Departments_Courses.

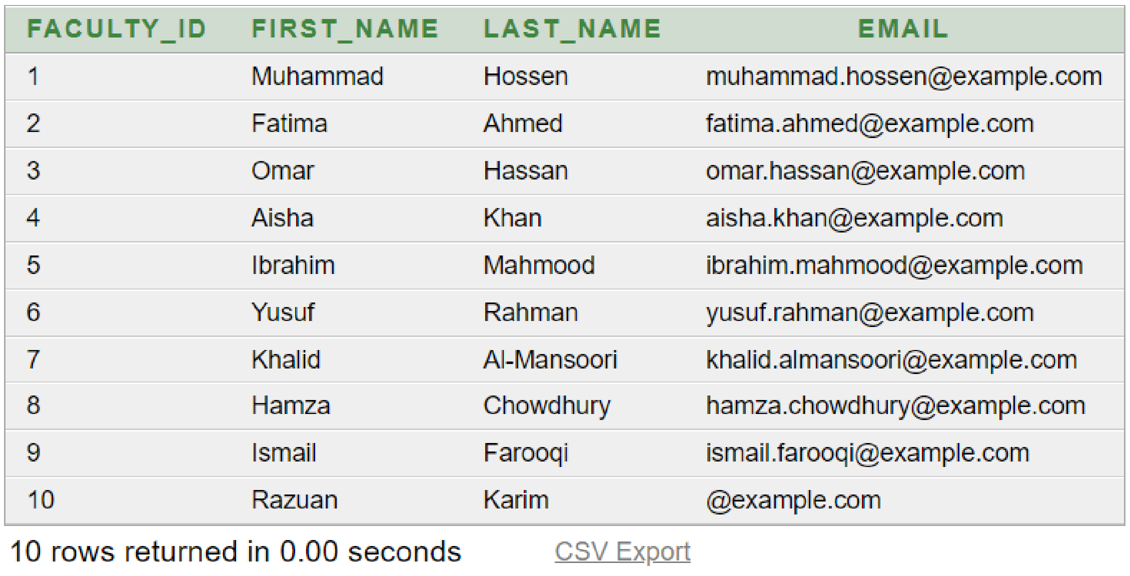
Value insertion of Table: Faculty



```
INSERT ALL
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(1, 'Muhammad','Hossen','muhammad.hossen@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(2, 'Fatima','Ahmed','fatima.ahmed@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(3, 'Omar','Hassan','omar.hassan@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(4, 'Aisha','Khan','aisha.khan@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(5, 'Ibrahim','Mahmood','ibrahim.mahmood@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(6, 'Yusuf','Rahman','yusuf.rahman@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(7, 'Khalid','Al-Mansoori','khalid.almansoori@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(8, 'Hamza','Chowdhury','hamza.chowdhury@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(9, 'Ismail','Farooqi','ismail.farooqi@example.com')
INTO Faculty (Faculty_ID, First_Name, Last_Name, Email) VALUES
(10, 'Razuan','Karim','@example.com')
SELECT * FROM dual;
```

```
10 row(s) inserted.
```

Fig 35: Value insertion command for table Faculty

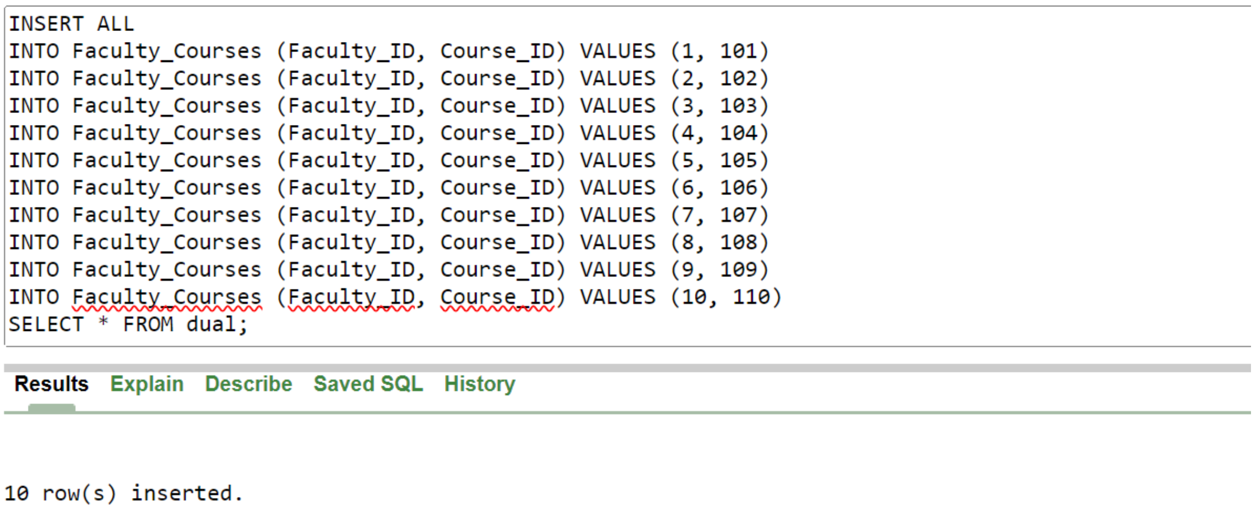


FACULTY_ID	FIRST_NAME	LAST_NAME	EMAIL
1	Muhammad	Hossen	muhammad.hossen@example.com
2	Fatima	Ahmed	fatima.ahmed@example.com
3	Omar	Hassan	omar.hassan@example.com
4	Aisha	Khan	aisha.khan@example.com
5	Ibrahim	Mahmood	ibrahim.mahmood@example.com
6	Yusuf	Rahman	yusuf.rahman@example.com
7	Khalid	Al-Mansoori	khalid.almansoori@example.com
8	Hamza	Chowdhury	hamza.chowdhury@example.com
9	Ismail	Farooqi	ismail.farooqi@example.com
10	Razuan	Karim	@example.com

10 rows returned in 0.00 seconds CSV Export

Fig 36: All data of table Faculty

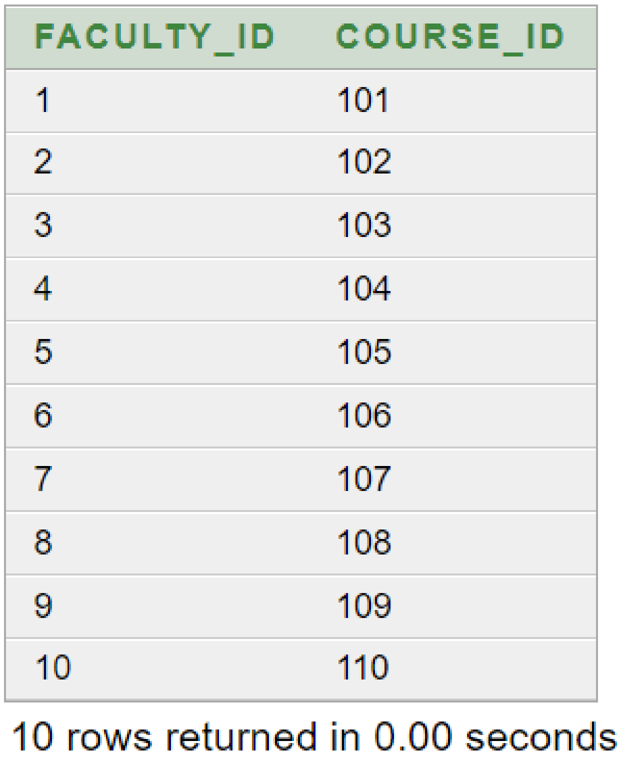
Value insertion of Table: Faculty_Courses



```
INSERT ALL
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (1, 101)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (2, 102)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (3, 103)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (4, 104)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (5, 105)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (6, 106)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (7, 107)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (8, 108)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (9, 109)
INTO Faculty_Courses (Faculty_ID, Course_ID) VALUES (10, 110)
SELECT * FROM dual;
```

```
10 row(s) inserted.
```

Fig 37: Value insertion command for table Faculty_Courses.

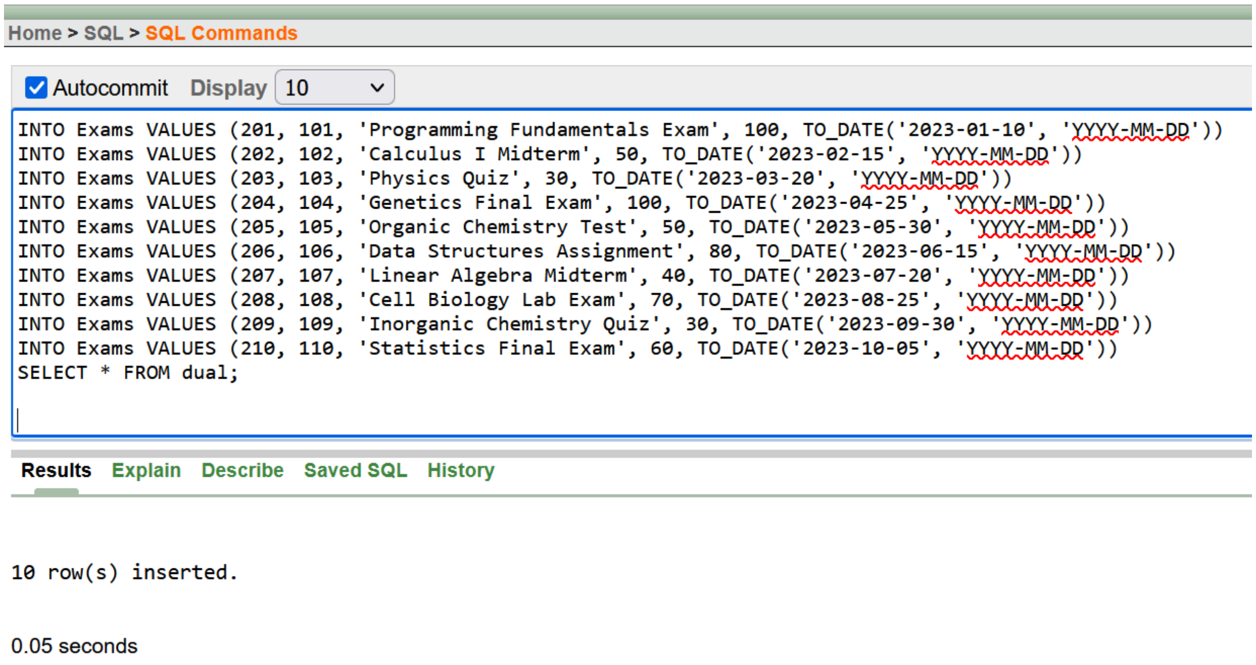


FACULTY_ID	COURSE_ID
1	101
2	102
3	103
4	104
5	105
6	106
7	107
8	108
9	109
10	110

10 rows returned in 0.00 seconds

Fig 38: All data of table Faculty_Courses

Value insertion of Table: Exams

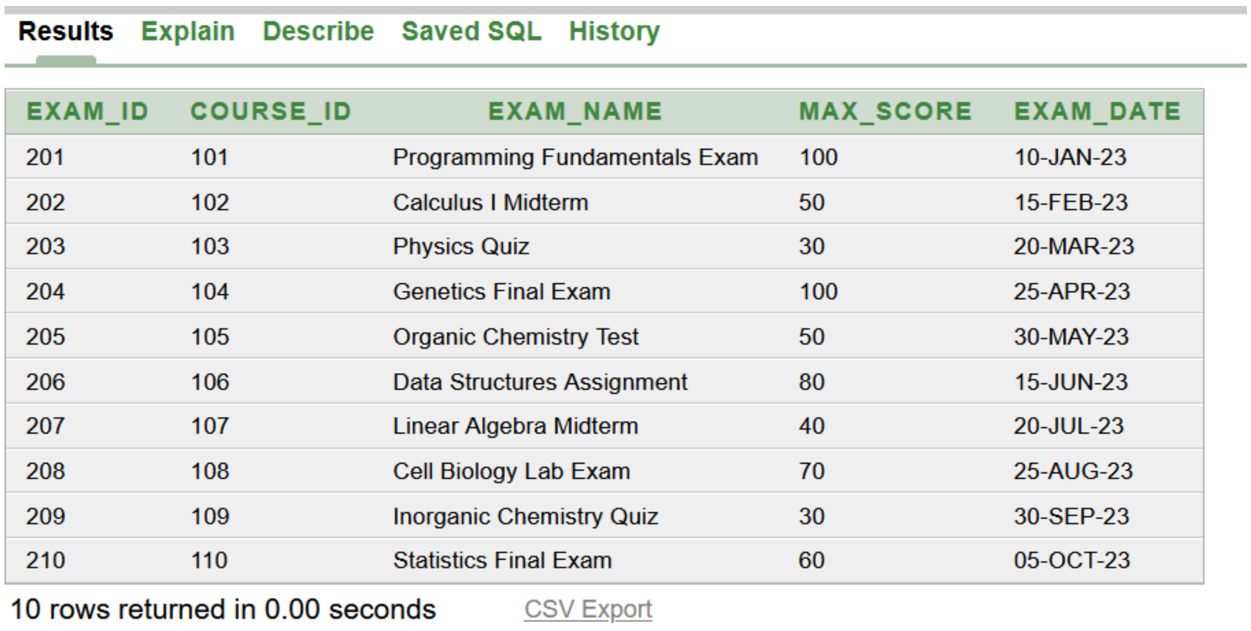

Home > SQL > SQL Commands
Autocommit Display 10
Results Explain Describe Saved SQL History

```
INTO Exams VALUES (201, 101, 'Programming Fundamentals Exam', 100, TO_DATE('2023-01-10', 'YYYY-MM-DD'))
INTO Exams VALUES (202, 102, 'Calculus I Midterm', 50, TO_DATE('2023-02-15', 'YYYY-MM-DD'))
INTO Exams VALUES (203, 103, 'Physics Quiz', 30, TO_DATE('2023-03-20', 'YYYY-MM-DD'))
INTO Exams VALUES (204, 104, 'Genetics Final Exam', 100, TO_DATE('2023-04-25', 'YYYY-MM-DD'))
INTO Exams VALUES (205, 105, 'Organic Chemistry Test', 50, TO_DATE('2023-05-30', 'YYYY-MM-DD'))
INTO Exams VALUES (206, 106, 'Data Structures Assignment', 80, TO_DATE('2023-06-15', 'YYYY-MM-DD'))
INTO Exams VALUES (207, 107, 'Linear Algebra Midterm', 40, TO_DATE('2023-07-20', 'YYYY-MM-DD'))
INTO Exams VALUES (208, 108, 'Cell Biology Lab Exam', 70, TO_DATE('2023-08-25', 'YYYY-MM-DD'))
INTO Exams VALUES (209, 109, 'Inorganic Chemistry Quiz', 30, TO_DATE('2023-09-30', 'YYYY-MM-DD'))
INTO Exams VALUES (210, 110, 'Statistics Final Exam', 60, TO_DATE('2023-10-05', 'YYYY-MM-DD'))
SELECT * FROM dual;
```

```
10 row(s) inserted.

0.05 seconds
```

Fig 39: Value insertion command for table Exams.

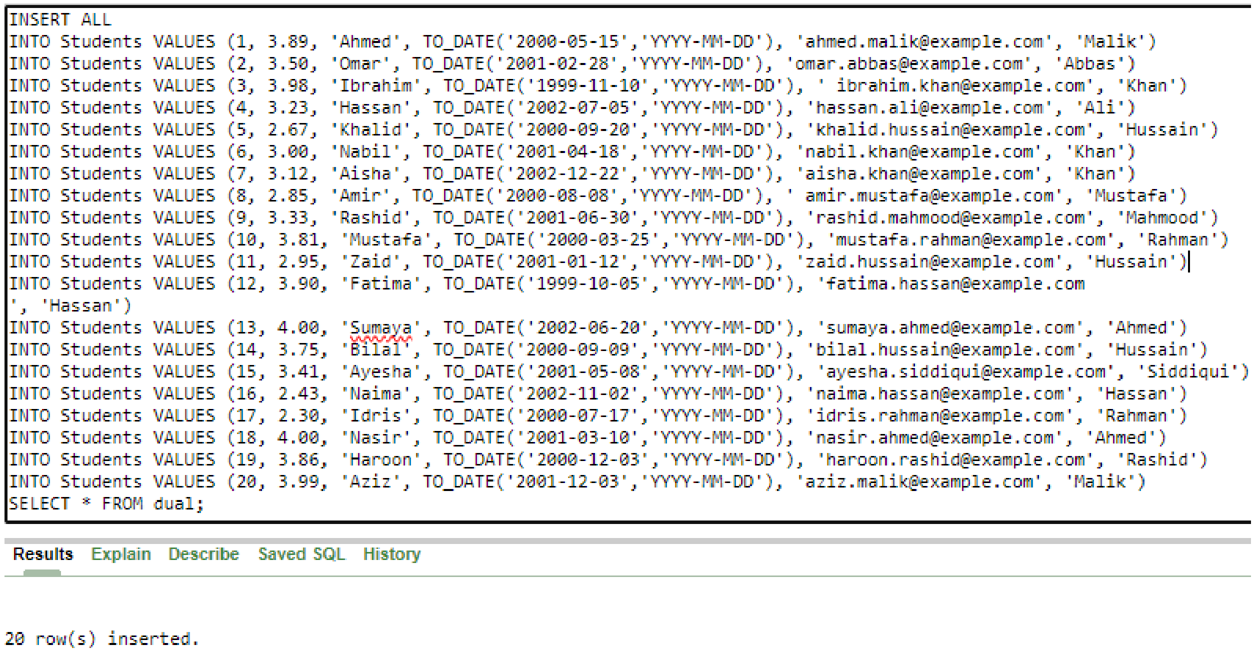


EXAM_ID	COURSE_ID	EXAM_NAME	MAX_SCORE	EXAM_DATE
201	101	Programming Fundamentals Exam	100	10-JAN-23
202	102	Calculus I Midterm	50	15-FEB-23
203	103	Physics Quiz	30	20-MAR-23
204	104	Genetics Final Exam	100	25-APR-23
205	105	Organic Chemistry Test	50	30-MAY-23
206	106	Data Structures Assignment	80	15-JUN-23
207	107	Linear Algebra Midterm	40	20-JUL-23
208	108	Cell Biology Lab Exam	70	25-AUG-23
209	109	Inorganic Chemistry Quiz	30	30-SEP-23
210	110	Statistics Final Exam	60	05-OCT-23

10 rows returned in 0.00 seconds

Fig 40: All data of table Exams.

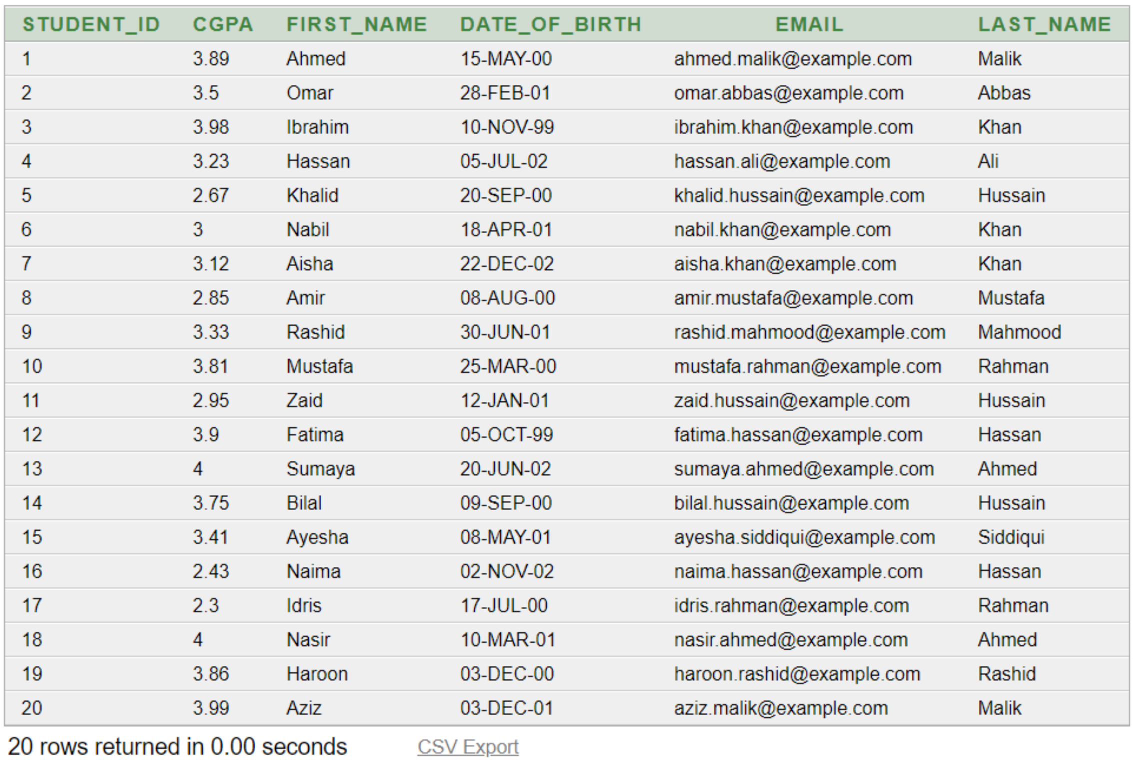
Value insertion of Table: Students



```
INSERT ALL
INTO Students VALUES (1, 3.89, 'Ahmed', TO_DATE('2000-05-15','YYYY-MM-DD'), 'ahmed.malik@example.com', 'Malik')
INTO Students VALUES (2, 3.50, 'Omar', TO_DATE('2001-02-28','YYYY-MM-DD'), 'omar.abbas@example.com', 'Abbas')
INTO Students VALUES (3, 3.98, 'Ibrahim', TO_DATE('1999-11-10','YYYY-MM-DD'), ' ibrahim.khan@example.com', 'Khan')
INTO Students VALUES (4, 3.23, 'Hassan', TO_DATE('2002-07-05','YYYY-MM-DD'), 'hassan.ali@example.com', 'Ali')
INTO Students VALUES (5, 2.67, 'Khalid', TO_DATE('2000-09-20','YYYY-MM-DD'), 'khalid.hussain@example.com', 'Hussain')
INTO Students VALUES (6, 3.00, 'Nabil', TO_DATE('2001-04-18','YYYY-MM-DD'), 'nabil.khan@example.com', 'Khan')
INTO Students VALUES (7, 3.12, 'Aisha', TO_DATE('2002-12-22','YYYY-MM-DD'), 'aisha.khan@example.com', 'Khan')
INTO Students VALUES (8, 2.85, 'Amir', TO_DATE('2000-08-08','YYYY-MM-DD'), ' amir.mustafa@example.com', 'Mustafa')
INTO Students VALUES (9, 3.33, 'Rashid', TO_DATE('2001-06-30','YYYY-MM-DD'), 'rashid.mahmood@example.com', 'Mahmood')
INTO Students VALUES (10, 3.81, 'Mustafa', TO_DATE('2000-03-25','YYYY-MM-DD'), 'mustafa.rahman@example.com', 'Rahman')
INTO Students VALUES (11, 2.95, 'Zaid', TO_DATE('2001-01-12','YYYY-MM-DD'), 'zaid.hussain@example.com', 'Hussain')
INTO Students VALUES (12, 3.90, 'Fatima', TO_DATE('1999-10-05','YYYY-MM-DD'), 'fatima.hassan@example.com
', 'Hassan')
INTO Students VALUES (13, 4.00, 'Sumaya', TO_DATE('2002-06-20','YYYY-MM-DD'), 'sumaya.ahmed@example.com', 'Ahmed')
INTO Students VALUES (14, 3.75, 'Bilal', TO_DATE('2000-09-09','YYYY-MM-DD'), 'bilal.hussain@example.com', 'Hussain')
INTO Students VALUES (15, 3.41, 'Ayesha', TO_DATE('2001-05-08','YYYY-MM-DD'), 'ayesha.siddiqui@example.com', 'Siddiqui')
INTO Students VALUES (16, 2.43, 'Naima', TO_DATE('2002-11-02','YYYY-MM-DD'), 'naima.hassan@example.com', 'Hassan')
INTO Students VALUES (17, 2.30, 'Idris', TO_DATE('2000-07-17','YYYY-MM-DD'), 'idris.rahman@example.com', 'Rahman')
INTO Students VALUES (18, 4.00, 'Nasir', TO_DATE('2001-03-10','YYYY-MM-DD'), 'nasir.ahmed@example.com', 'Ahmed')
INTO Students VALUES (19, 3.86, 'Haroon', TO_DATE('2000-12-03','YYYY-MM-DD'), 'haroon.rashid@example.com', 'Rashid')
INTO Students VALUES (20, 3.99, 'Aziz', TO_DATE('2001-12-03','YYYY-MM-DD'), 'aziz.malik@example.com', 'Malik')
SELECT * FROM dual;
```

```
20 row(s) inserted.
```

Fig 41: Value insertion command for table Students.

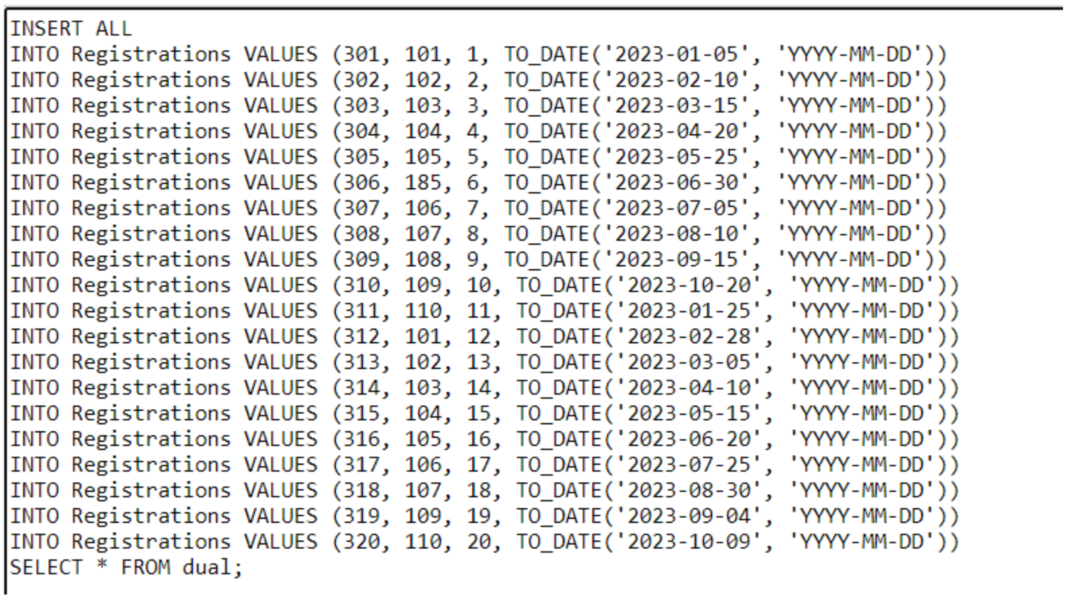


STUDENT_ID	CGPA	FIRST_NAME	DATE_OF_BIRTH	EMAIL	LAST_NAME
1	3.89	Ahmed	15-MAY-00	ahmed.malik@example.com	Malik
2	3.5	Omar	28-FEB-01	omar.abbas@example.com	Abbas
3	3.98	Ibrahim	10-NOV-99	ibrahim.khan@example.com	Khan
4	3.23	Hassan	05-JUL-02	hassan.ali@example.com	Ali
5	2.67	Khalid	20-SEP-00	khalid.hussain@example.com	Hussain
6	3	Nabil	18-APR-01	nabil.khan@example.com	Khan
7	3.12	Aisha	22-DEC-02	aisha.khan@example.com	Khan
8	2.85	Amir	08-AUG-00	amir.mustafa@example.com	Mustafa
9	3.33	Rashid	30-JUN-01	rashid.mahmood@example.com	Mahmood
10	3.81	Mustafa	25-MAR-00	mustafa.rahman@example.com	Rahman
11	2.95	Zaid	12-JAN-01	zaid.hussain@example.com	Hussain
12	3.9	Fatima	05-OCT-99	fatima.hassan@example.com	Hassan
13	4	Sumaya	20-JUN-02	sumaya.ahmed@example.com	Ahmed
14	3.75	Bilal	09-SEP-00	bilal.hussain@example.com	Hussain
15	3.41	Ayesha	08-MAY-01	ayesha.siddiqui@example.com	Siddiqui
16	2.43	Naima	02-NOV-02	naima.hassan@example.com	Hassan
17	2.3	Idris	17-JUL-00	idris.rahman@example.com	Rahman
18	4	Nasir	10-MAR-01	nasir.ahmed@example.com	Ahmed
19	3.86	Haroon	03-DEC-00	haroon.rashid@example.com	Rashid
20	3.99	Aziz	03-DEC-01	aziz.malik@example.com	Malik

20 rows returned in 0.00 seconds

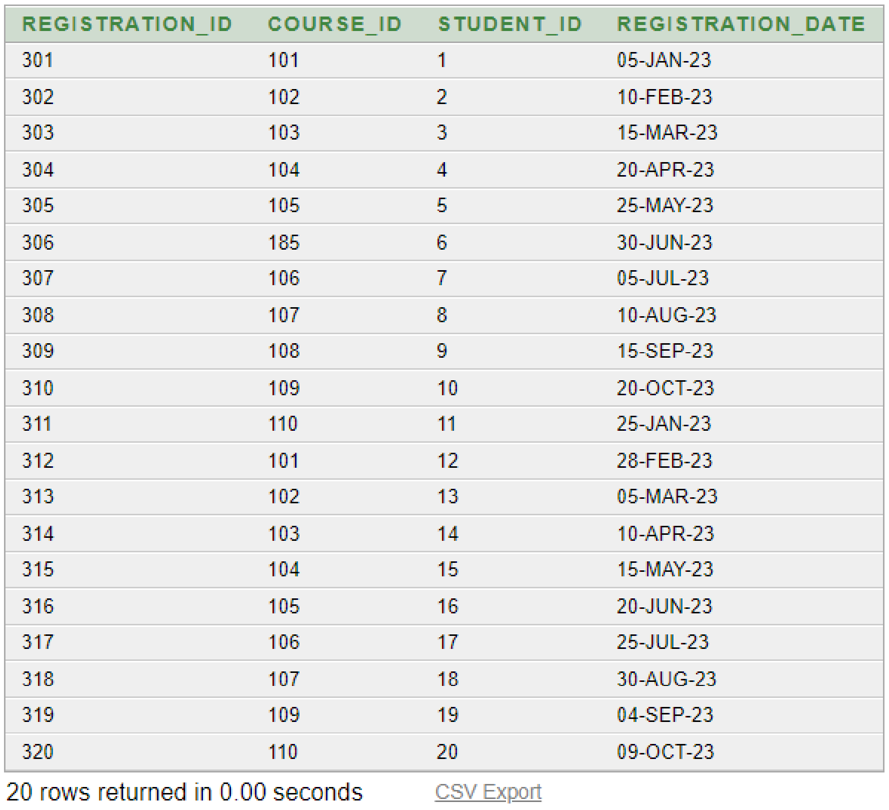
Fig 42: All data of table Students.

Value insertion of Table: Registrations



```
INSERT ALL
INTO Registrations VALUES (301, 101, 1, TO_DATE('2023-01-05', 'YYYY-MM-DD'))
INTO Registrations VALUES (302, 102, 2, TO_DATE('2023-02-10', 'YYYY-MM-DD'))
INTO Registrations VALUES (303, 103, 3, TO_DATE('2023-03-15', 'YYYY-MM-DD'))
INTO Registrations VALUES (304, 104, 4, TO_DATE('2023-04-20', 'YYYY-MM-DD'))
INTO Registrations VALUES (305, 105, 5, TO_DATE('2023-05-25', 'YYYY-MM-DD'))
INTO Registrations VALUES (306, 185, 6, TO_DATE('2023-06-30', 'YYYY-MM-DD'))
INTO Registrations VALUES (307, 106, 7, TO_DATE('2023-07-05', 'YYYY-MM-DD'))
INTO Registrations VALUES (308, 107, 8, TO_DATE('2023-08-10', 'YYYY-MM-DD'))
INTO Registrations VALUES (309, 108, 9, TO_DATE('2023-09-15', 'YYYY-MM-DD'))
INTO Registrations VALUES (310, 109, 10, TO_DATE('2023-10-20', 'YYYY-MM-DD'))
INTO Registrations VALUES (311, 110, 11, TO_DATE('2023-01-25', 'YYYY-MM-DD'))
INTO Registrations VALUES (312, 101, 12, TO_DATE('2023-02-28', 'YYYY-MM-DD'))
INTO Registrations VALUES (313, 102, 13, TO_DATE('2023-03-05', 'YYYY-MM-DD'))
INTO Registrations VALUES (314, 103, 14, TO_DATE('2023-04-10', 'YYYY-MM-DD'))
INTO Registrations VALUES (315, 104, 15, TO_DATE('2023-05-15', 'YYYY-MM-DD'))
INTO Registrations VALUES (316, 105, 16, TO_DATE('2023-06-20', 'YYYY-MM-DD'))
INTO Registrations VALUES (317, 106, 17, TO_DATE('2023-07-25', 'YYYY-MM-DD'))
INTO Registrations VALUES (318, 107, 18, TO_DATE('2023-08-30', 'YYYY-MM-DD'))
INTO Registrations VALUES (319, 109, 19, TO_DATE('2023-09-04', 'YYYY-MM-DD'))
INTO Registrations VALUES (320, 110, 20, TO_DATE('2023-10-09', 'YYYY-MM-DD'))
SELECT * FROM dual;
```

Fig 43: Value insertion command for table Registration

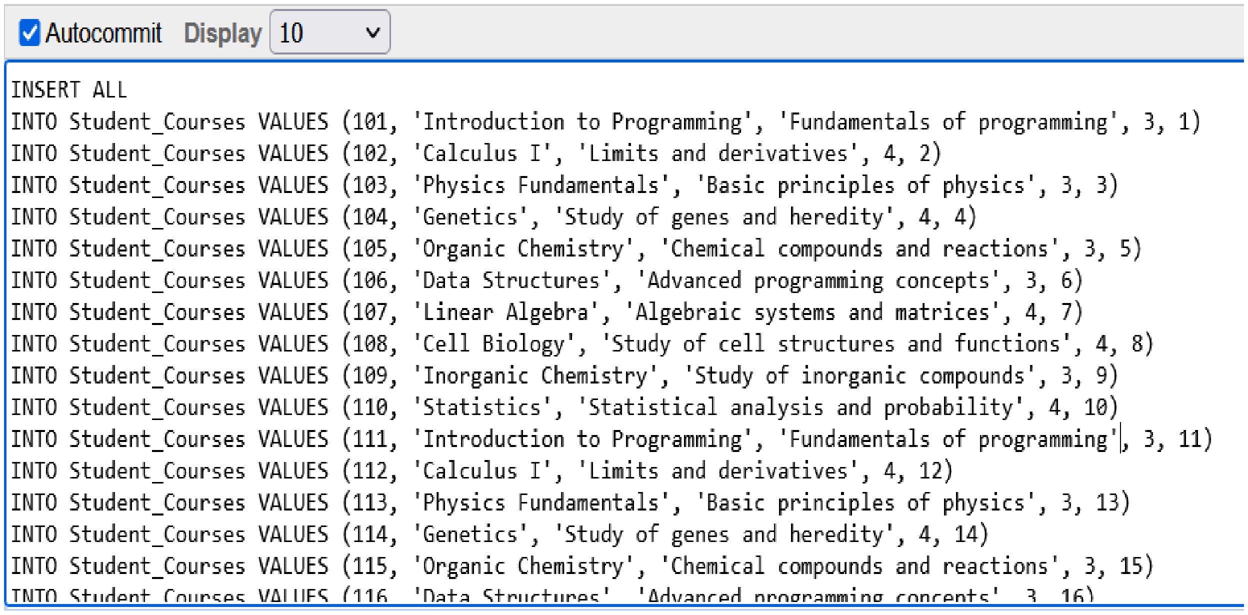


REGISTRATION_ID	COURSE_ID	STUDENT_ID	REGISTRATION_DATE
301	101	1	05-JAN-23
302	102	2	10-FEB-23
303	103	3	15-MAR-23
304	104	4	20-APR-23
305	105	5	25-MAY-23
306	185	6	30-JUN-23
307	106	7	05-JUL-23
308	107	8	10-AUG-23
309	108	9	15-SEP-23
310	109	10	20-OCT-23
311	110	11	25-JAN-23
312	101	12	28-FEB-23
313	102	13	05-MAR-23
314	103	14	10-APR-23
315	104	15	15-MAY-23
316	105	16	20-JUN-23
317	106	17	25-JUL-23
318	107	18	30-AUG-23
319	109	19	04-SEP-23
320	110	20	09-OCT-23

20 rows returned in 0.00 seconds CSV Export

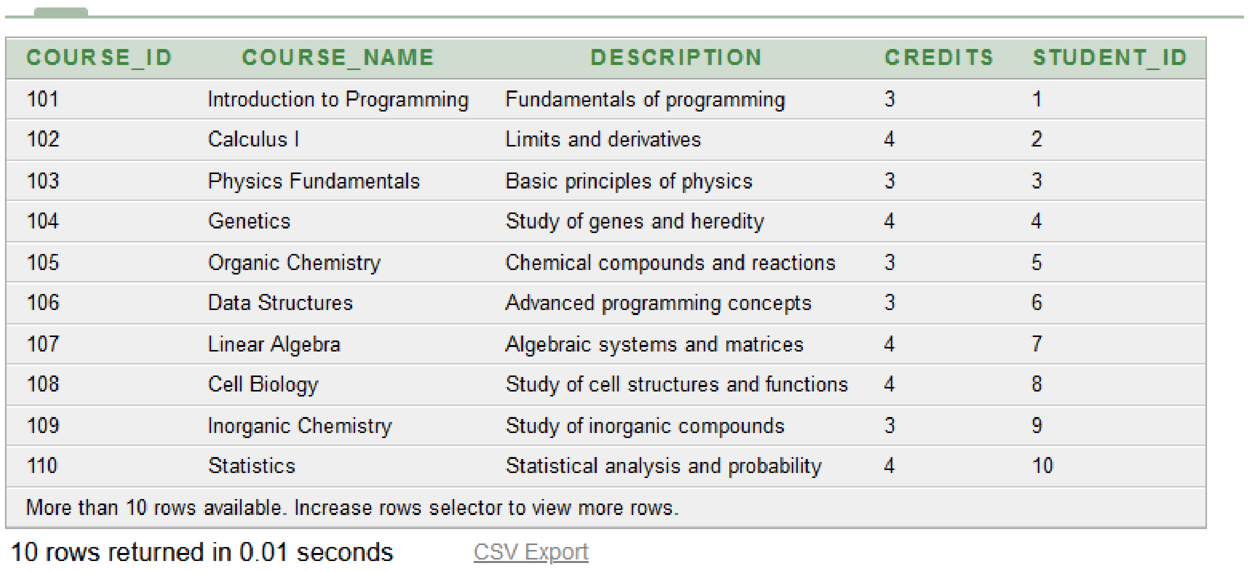
Fig 44: All data of table Registration

Value insertion of Table: Student_Courses



```
INSERT ALL
INTO Student_Courses VALUES (101, 'Introduction to Programming', 'Fundamentals of programming', 3, 1)
INTO Student_Courses VALUES (102, 'Calculus I', 'Limits and derivatives', 4, 2)
INTO Student_Courses VALUES (103, 'Physics Fundamentals', 'Basic principles of physics', 3, 3)
INTO Student_Courses VALUES (104, 'Genetics', 'Study of genes and heredity', 4, 4)
INTO Student_Courses VALUES (105, 'Organic Chemistry', 'Chemical compounds and reactions', 3, 5)
INTO Student_Courses VALUES (106, 'Data Structures', 'Advanced programming concepts', 3, 6)
INTO Student_Courses VALUES (107, 'Linear Algebra', 'Algebraic systems and matrices', 4, 7)
INTO Student_Courses VALUES (108, 'Cell Biology', 'Study of cell structures and functions', 4, 8)
INTO Student_Courses VALUES (109, 'Inorganic Chemistry', 'Study of inorganic compounds', 3, 9)
INTO Student_Courses VALUES (110, 'Statistics', 'Statistical analysis and probability', 4, 10)
INTO Student_Courses VALUES (111, 'Introduction to Programming', 'Fundamentals of programming', 3, 11)
INTO Student_Courses VALUES (112, 'Calculus I', 'Limits and derivatives', 4, 12)
INTO Student_Courses VALUES (113, 'Physics Fundamentals', 'Basic principles of physics', 3, 13)
INTO Student_Courses VALUES (114, 'Genetics', 'Study of genes and heredity', 4, 14)
INTO Student_Courses VALUES (115, 'Organic Chemistry', 'Chemical compounds and reactions', 3, 15)
INTO Student_Courses VALUES (116, 'Data Structures', 'Advanced programming concepts', 3, 16)
```

Fig 45: Value insertion command for Student_Courses.



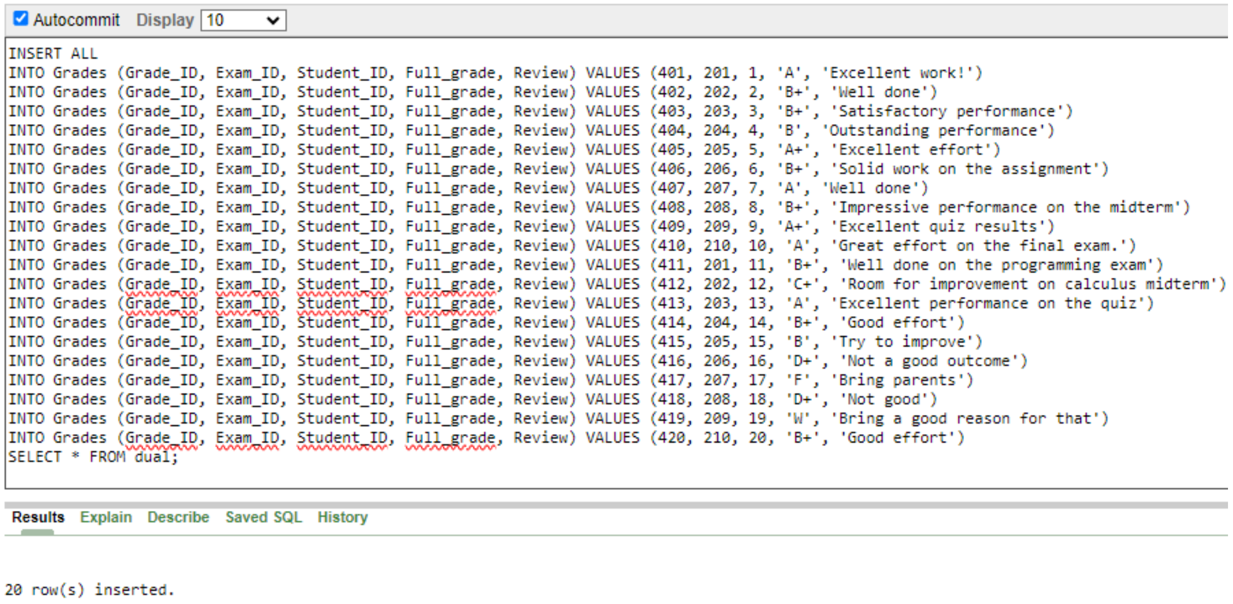
COURSE_ID	COURSE_NAME	DESCRIPTION	CREDITS	STUDENT_ID
101	Introduction to Programming	Fundamentals of programming	3	1
102	Calculus I	Limits and derivatives	4	2
103	Physics Fundamentals	Basic principles of physics	3	3
104	Genetics	Study of genes and heredity	4	4
105	Organic Chemistry	Chemical compounds and reactions	3	5
106	Data Structures	Advanced programming concepts	3	6
107	Linear Algebra	Algebraic systems and matrices	4	7
108	Cell Biology	Study of cell structures and functions	4	8
109	Inorganic Chemistry	Study of inorganic compounds	3	9
110	Statistics	Statistical analysis and probability	4	10

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds

Fig 46: All data of table Student_Courses.

Value insertion of Table: Grades



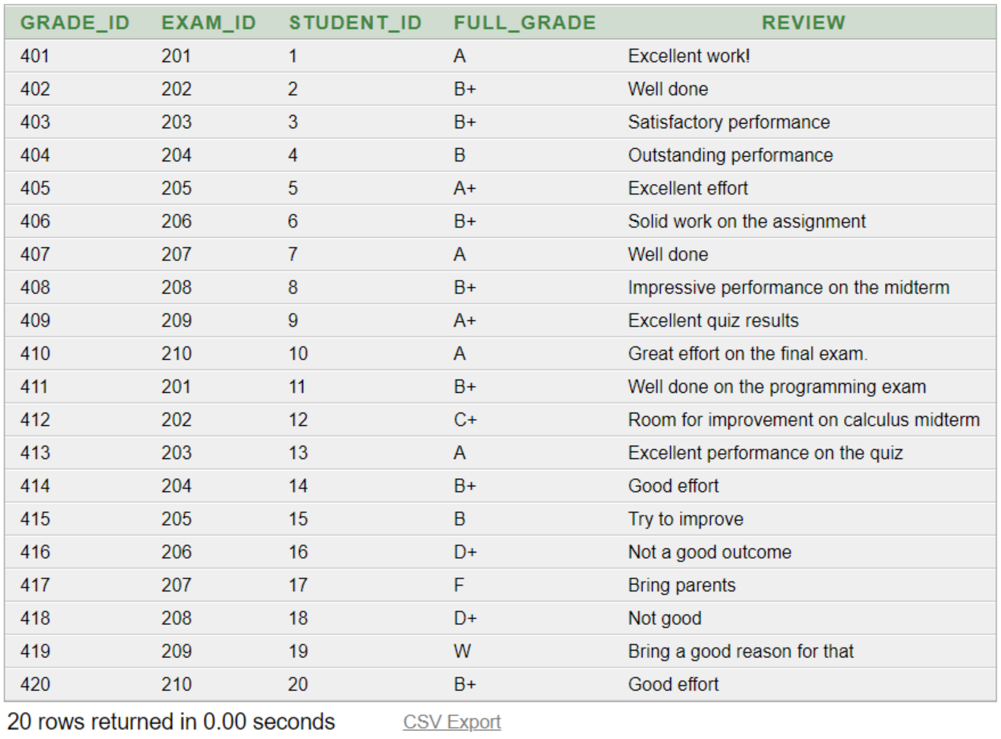
Autocommit Display 10

```
INSERT ALL
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (401, 201, 1, 'A', 'Excellent work!')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (402, 202, 2, 'B+', 'Well done')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (403, 203, 3, 'B+', 'Satisfactory performance')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (404, 204, 4, 'B', 'Outstanding performance')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (405, 205, 5, 'A+', 'Excellent effort')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (406, 206, 6, 'B+', 'Solid work on the assignment')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (407, 207, 7, 'A', 'Well done')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (408, 208, 8, 'B+', 'Impressive performance on the midterm')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (409, 209, 9, 'A+', 'Excellent quiz results')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (410, 210, 10, 'A', 'Great effort on the final exam.')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (411, 201, 11, 'B+', 'Well done on the programming exam')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (412, 202, 12, 'C+', 'Room for improvement on calculus midterm')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (413, 203, 13, 'A', 'Excellent performance on the quiz')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (414, 204, 14, 'B+', 'Good effort')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (415, 205, 15, 'B', 'Try to improve')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (416, 206, 16, 'D+', 'Not a good outcome')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (417, 207, 17, 'F', 'Bring parents')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (418, 208, 18, 'D+', 'Not good')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (419, 209, 19, 'W', 'Bring a good reason for that')
INTO Grades (Grade_ID, Exam_ID, Student_ID, Full_grade, Review) VALUES (420, 210, 20, 'B+', 'Good effort')
SELECT * FROM dual;
```

Results Explain Describe Saved SQL History

```
20 row(s) inserted.
```

Fig 47: Value insertion command for Grades.



GRADE_ID	EXAM_ID	STUDENT_ID	FULL_GRADE	REVIEW
401	201	1	A	Excellent work!
402	202	2	B+	Well done
403	203	3	B+	Satisfactory performance
404	204	4	B	Outstanding performance
405	205	5	A+	Excellent effort
406	206	6	B+	Solid work on the assignment
407	207	7	A	Well done
408	208	8	B+	Impressive performance on the midterm
409	209	9	A+	Excellent quiz results
410	210	10	A	Great effort on the final exam.
411	201	11	B+	Well done on the programming exam
412	202	12	C+	Room for improvement on calculus midterm
413	203	13	A	Excellent performance on the quiz
414	204	14	B+	Good effort
415	205	15	B	Try to improve
416	206	16	D+	Not a good outcome
417	207	17	F	Bring parents
418	208	18	D+	Not good
419	209	19	W	Bring a good reason for that
420	210	20	B+	Good effort

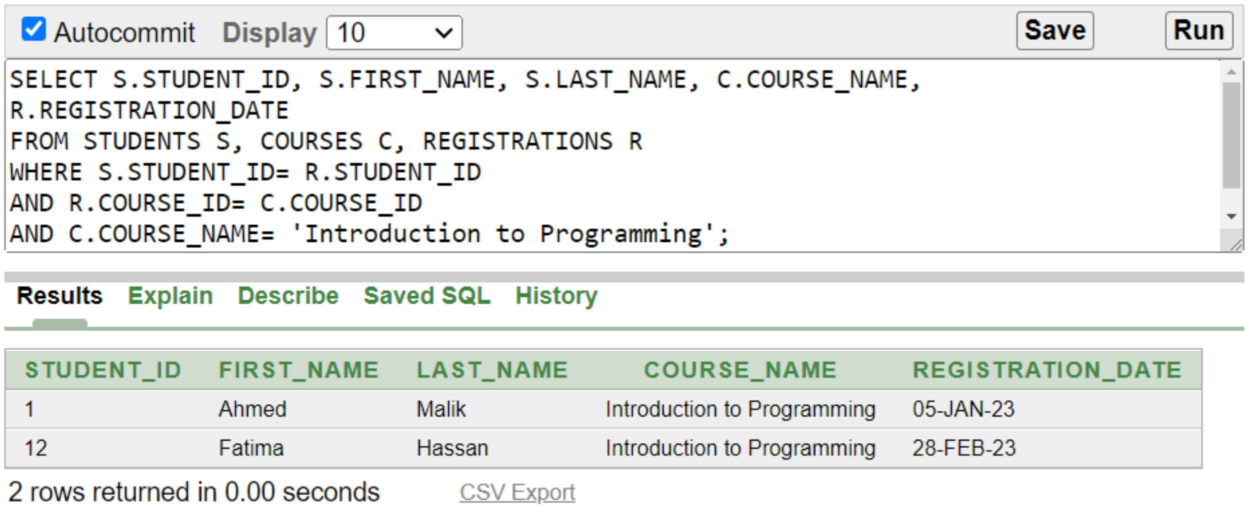
20 rows returned in 0.00 seconds CSV Export

Fig 48: All data of table Grades.

QUERY TEST

➔ Simple Query

Question 1: Show the details of students registered in the course 'Introduction to Programming' along with their names and Registration dates.



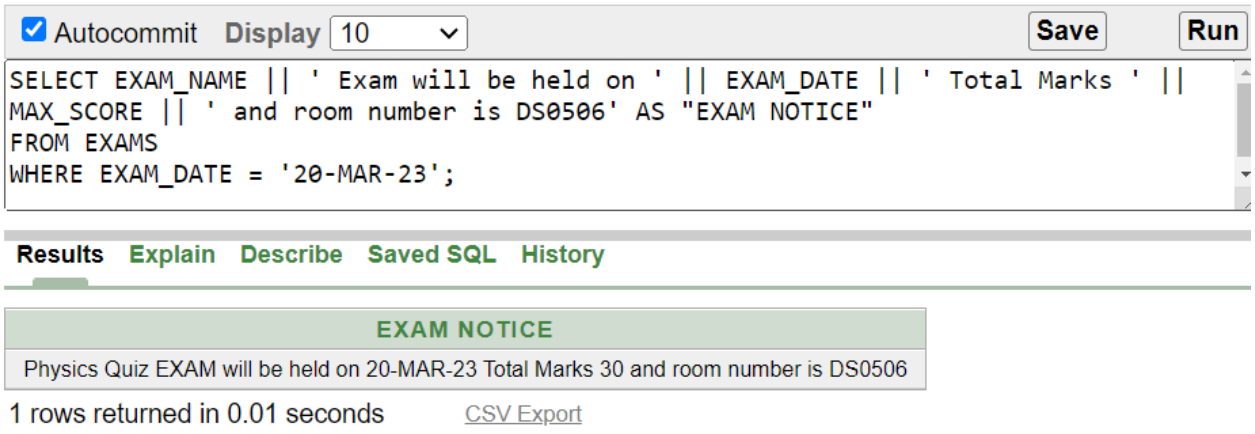
```
SELECT S.STUDENT_ID, S.FIRST_NAME, S.LAST_NAME, C.COURSE_NAME,
R.REGISTRATION_DATE
FROM STUDENTS S, COURSES C, REGISTRATIONS R
WHERE S.STUDENT_ID= R.STUDENT_ID
AND R.COURSE_ID= C.COURSE_ID
AND C.COURSE_NAME= 'Introduction to Programming';
```

STUDENT_ID	FIRST_NAME	LAST_NAME	COURSE_NAME	REGISTRATION_DATE
1	Ahmed	Malik	Introduction to Programming	05-JAN-23
12	Fatima	Hassan	Introduction to Programming	28-FEB-23

2 rows returned in 0.00 seconds

Fig 49: Command and Result for Simple query.

Question 2: Display the details of the exam scheduled for '20-MAR-2023' from the "Exams" Table. Label the resulting column as "Exam Notice." The output should include the exam name followed by "exam will be held on," (the exam date), "and exam total mark is," (the exam mark), and the exam room number.



```
SELECT EXAM_NAME || ' Exam will be held on ' || EXAM_DATE || ' Total Marks ' ||
MAX_SCORE || ' and room number is DS0506' AS "EXAM NOTICE"
FROM EXAMS
WHERE EXAM_DATE = '20-MAR-23';
```

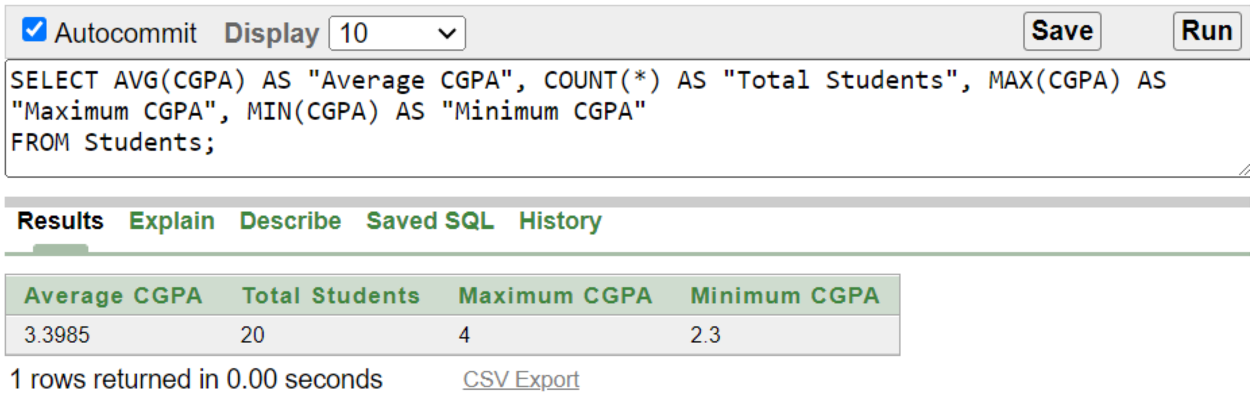
EXAM NOTICE
Physics Quiz EXAM will be held on 20-MAR-23 Total Marks 30 and room number is DS0506

1 rows returned in 0.01 seconds

Fig 50: Command and Result for Simple query.

➔ Aggregate Query

Question 1: Show the average CGPA of students and the total number of students. Also show maximum and minimum CGPA of this students.



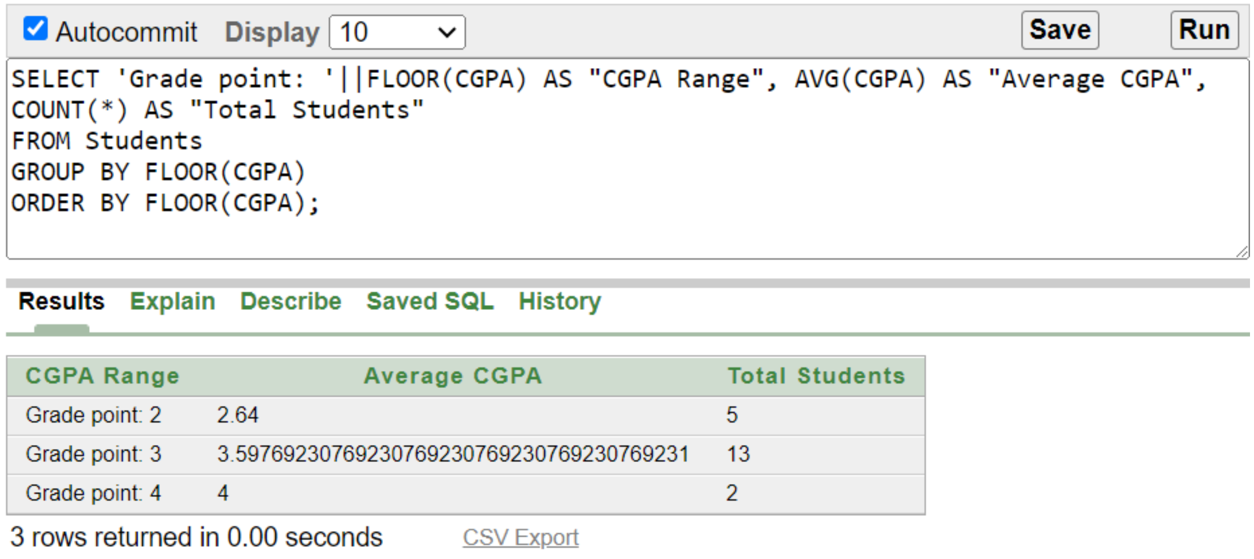
```
SELECT AVG(CGPA) AS "Average CGPA", COUNT(*) AS "Total Students", MAX(CGPA) AS
"Maximum CGPA", MIN(CGPA) AS "Minimum CGPA"
FROM Students;
```

Average CGPA	Total Students	Maximum CGPA	Minimum CGPA
3.3985	20	4	2.3

```
1 rows returned in 0.00 seconds
```

Fig 51: Command and Result for Aggregate query.

Question 2: Show the average CGPA of students and the total number of students according to grade point of CGPA. Level grade point of CGPA as 'CGPA Range'.



```
SELECT 'Grade point: '||FLOOR(CGPA) AS "CGPA Range", AVG(CGPA) AS "Average CGPA",
COUNT(*) AS "Total Students"
FROM Students
GROUP BY FLOOR(CGPA)
ORDER BY FLOOR(CGPA);
```

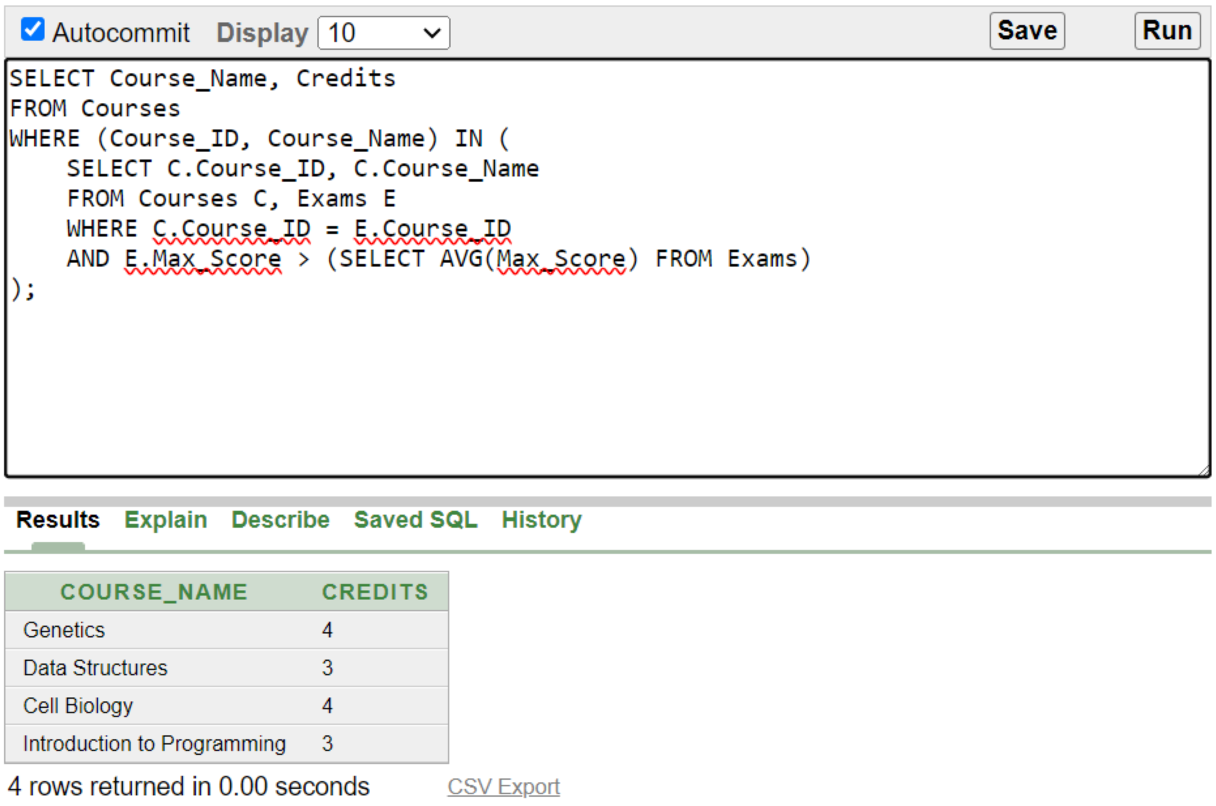
CGPA Range	Average CGPA	Total Students
Grade point: 2	2.64	5
Grade point: 3	3.5976923076923076923076923076923076923	13
Grade point: 4	4	2

```
3 rows returned in 0.00 seconds
```

Fig 52: Command and Result for Aggregate query.

➔ Single-Row Subquery

Question 1: Display the course names and credits for courses where the maximum exam score is greater than the average maximum score across all exams.



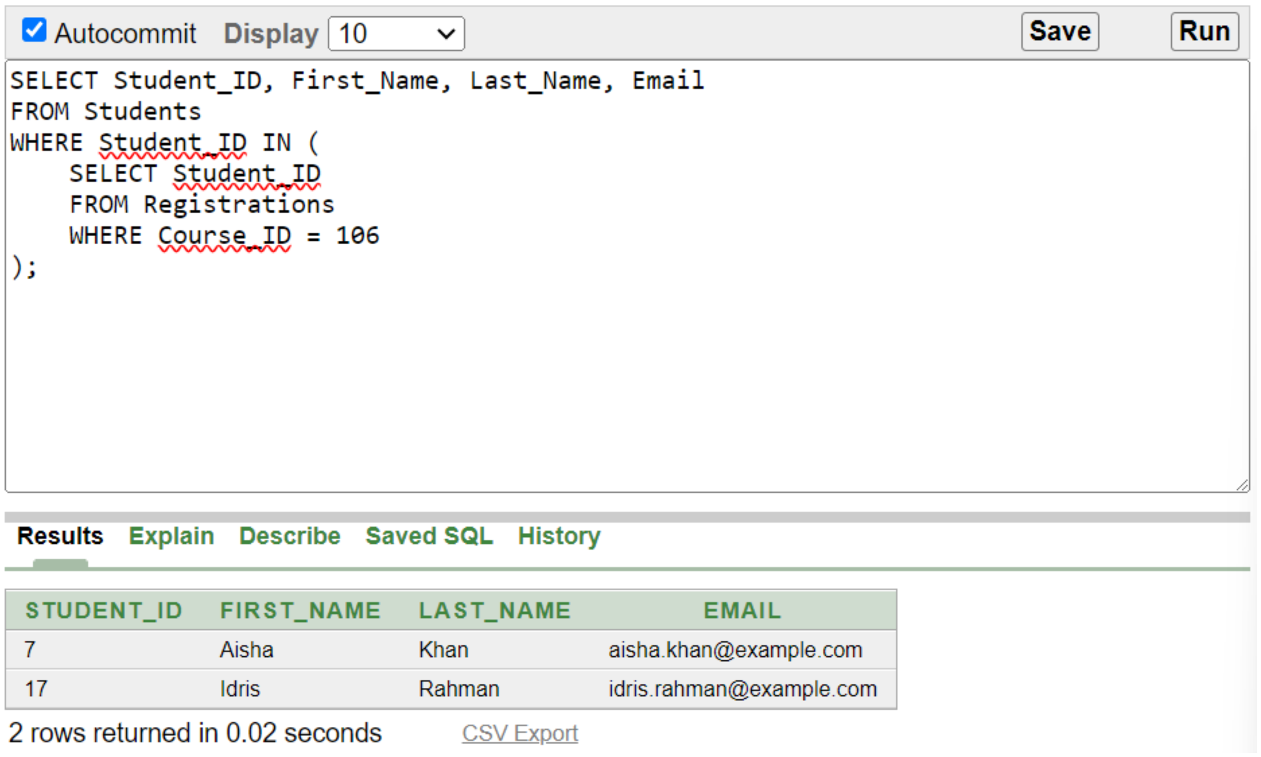
```
SELECT Course_Name, Credits
FROM Courses
WHERE (Course_ID, Course_Name) IN (
    SELECT C.Course_ID, C.Course_Name
    FROM Courses C, Exams E
    WHERE C.Course_ID = E.Course_ID
    AND E.Max_Score > (SELECT AVG(Max_Score) FROM Exams)
);
```

COURSE_NAME	CREDITS
Genetics	4
Data Structures	3
Cell Biology	4
Introduction to Programming	3

4 rows returned in 0.00 seconds

Fig 53: Command and Results for Single-Row Subquery

Question 2: Retrieve the student_ID, FIRST_NAME and email addresses from the 'Students' table. Identify students who are either enrolled in a course with course_ID: 106 according to the 'Enrollments' table.



Display 10

```
SELECT Student_ID, First_Name, Last_Name, Email
FROM Students
WHERE Student_ID IN (
    SELECT Student_ID
    FROM Registrations
    WHERE Course_ID = 106
);
```

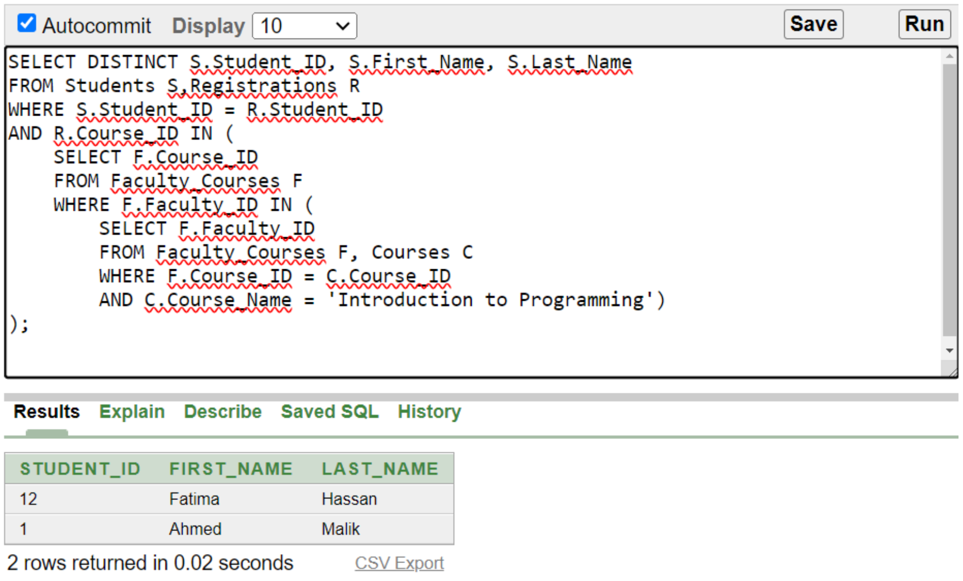
STUDENT_ID	FIRST_NAME	LAST_NAME	EMAIL
7	Aisha	Khan	aisha.khan@example.com
17	Idris	Rahman	idris.rahman@example.com

2 rows returned in 0.02 seconds

Fig 54: Command and Results for Single-Row Subquery

➔ Multiple-Row Subquery

Question 1: Display the student names and IDs for students who are registered in courses with the same faculty who teach 'Introduction to Programming.'



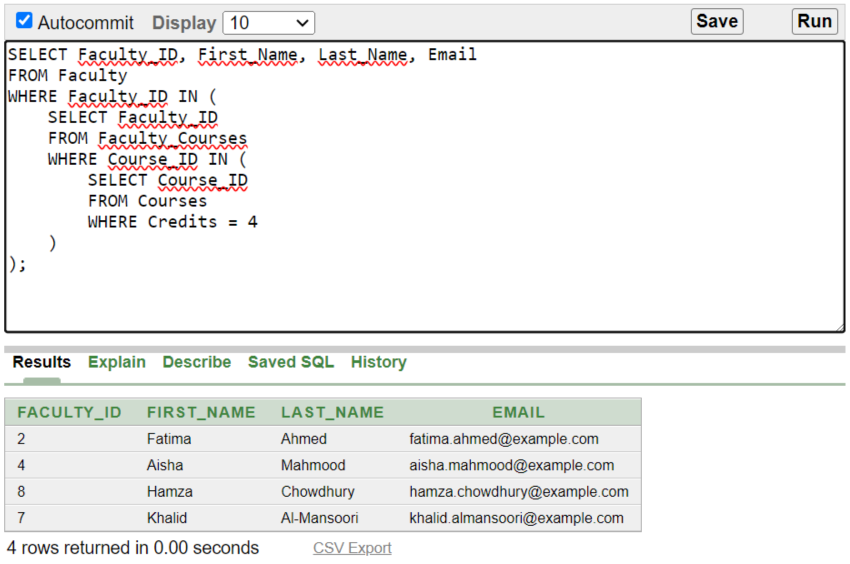
```
SELECT DISTINCT S.Student_ID, S.First_Name, S.Last_Name
FROM Students S,Registrations R
WHERE S.Student_ID = R.Student_ID
AND R.Course_ID IN (
    SELECT F.Course_ID
    FROM Faculty_Courses F
    WHERE F.Faculty_ID IN (
        SELECT F.Faculty_ID
        FROM Faculty_Courses F, Courses C
        WHERE F.Course_ID = C.Course_ID
        AND C.Course_Name = 'Introduction to Programming')
);
```

STUDENT_ID	FIRST_NAME	LAST_NAME
12	Fatima	Hassan
1	Ahmed	Malik

2 rows returned in 0.02 seconds

Fig 55: Command and Results for Multiple-Row Subquery

Question 2: Retrieve the ID, name, email, from the 'Instructors' table. Identify faculty members whose IDs match those found in the result of a subquery. This subquery finds IDs from the 'Instructor_Courses' table where the associated course IDs match any course ID from another subquery. The second subquery selects course IDs from the 'course' table where the course creditis 4.



```
SELECT Faculty_ID, First_Name, Last_Name, Email
FROM Faculty
WHERE Faculty_ID IN (
    SELECT Faculty_ID
    FROM Faculty_Courses
    WHERE Course_ID IN (
        SELECT Course_ID
        FROM Courses
        WHERE Credits = 4
    )
);
```

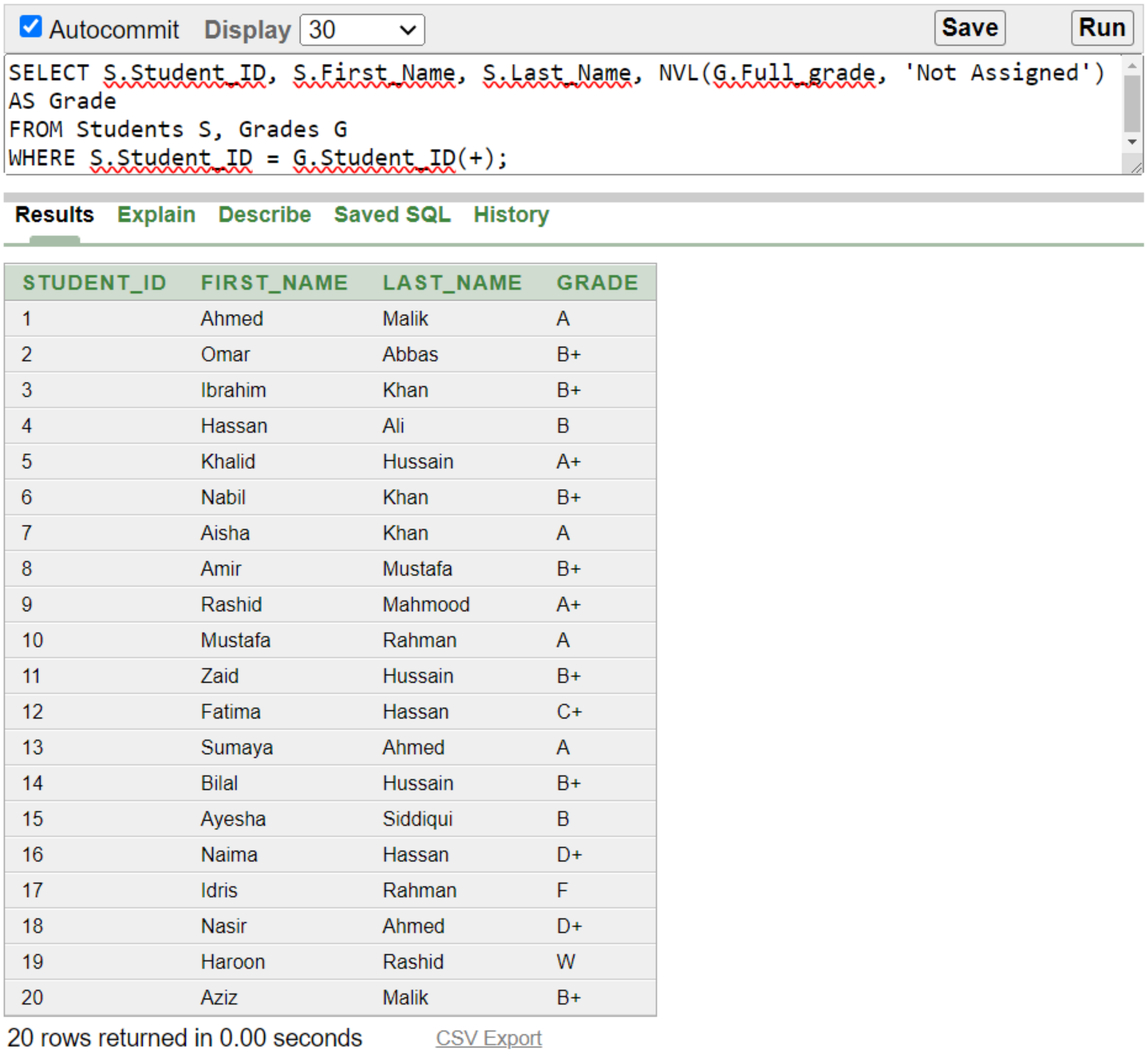
FACULTY_ID	FIRST_NAME	LAST_NAME	EMAIL
2	Fatima	Ahmed	fatima.ahmed@example.com
4	Aisha	Mahmood	aisha.mahmood@example.com
8	Hamza	Chowdhury	hamza.chowdhury@example.com
7	Khalid	Al-Mansoori	khalid.almansoori@example.com

4 rows returned in 0.00 seconds

Fig 56: Command and Results for Multiple-Row Subquery

➔ Joining - Outer Join

Question 1: Retrieve the student details and their corresponding grades using an outer join. Display the student ID, first name, last name, and grade value (if available). In case a student does not have an assigned grade, indicate it as 'Not Assigned'. Utilize a left outer join between the 'Students' and 'Grades' tables, linking records based on the 'Student_ID' column.



```
SELECT S.Student_ID, S.First_Name, S.Last_Name, NVL(G.Full_grade, 'Not Assigned')
AS Grade
FROM Students S, Grades G
WHERE S.Student_ID = G.Student_ID(+);
```

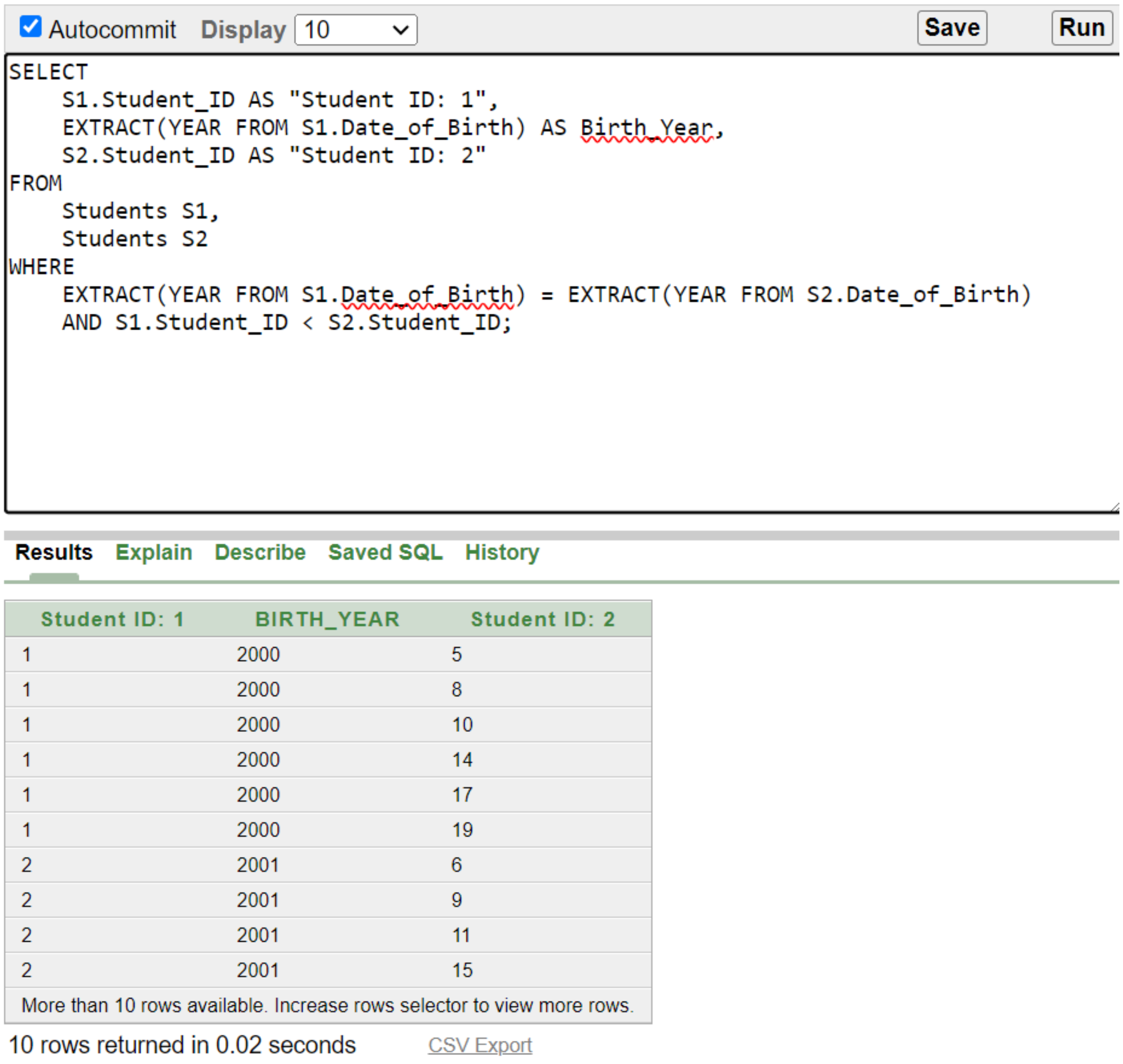
STUDENT_ID	FIRST_NAME	LAST_NAME	GRADE
1	Ahmed	Malik	A
2	Omar	Abbas	B+
3	Ibrahim	Khan	B+
4	Hassan	Ali	B
5	Khalid	Hussain	A+
6	Nabil	Khan	B+
7	Aisha	Khan	A
8	Amir	Mustafa	B+
9	Rashid	Mahmood	A+
10	Mustafa	Rahman	A
11	Zaid	Hussain	B+
12	Fatima	Hassan	C+
13	Sumaya	Ahmed	A
14	Bilal	Hussain	B+
15	Ayesha	Siddiqui	B
16	Naima	Hassan	D+
17	Idris	Rahman	F
18	Nasir	Ahmed	D+
19	Haroon	Rashid	W
20	Aziz	Malik	B+

20 rows returned in 0.00 seconds

Fig 57: Command and Results for Joining – Outer Join.

➔ Self-Joining

Question 1: Retrieve pairs of students who share the same date of birth year but have different student IDs.



```
SELECT
    S1.Student_ID AS "Student ID: 1",
    EXTRACT(YEAR FROM S1.Date_of_Birth) AS Birth_Year,
    S2.Student_ID AS "Student ID: 2"
FROM
    Students S1,
    Students S2
WHERE
    EXTRACT(YEAR FROM S1.Date_of_Birth) = EXTRACT(YEAR FROM S2.Date_of_Birth)
    AND S1.Student_ID < S2.Student_ID;
```

Student ID: 1	BIRTH_YEAR	Student ID: 2
1	2000	5
1	2000	8
1	2000	10
1	2000	14
1	2000	17
1	2000	19
2	2001	6
2	2001	9
2	2001	11
2	2001	15

More than 10 rows available. Increase rows selector to view more rows.

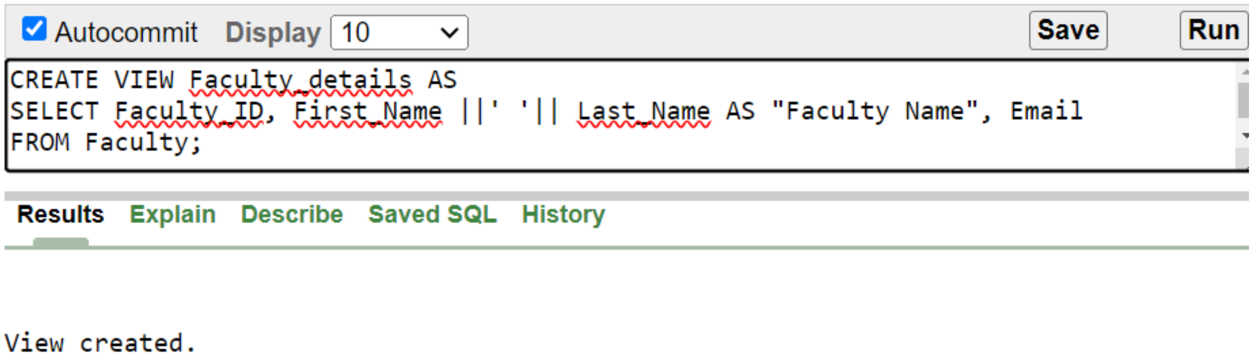
10 rows returned in 0.02 seconds

Fig 58: Command and Results for Self-Joining.

➔ View

1. Simple view

Question1: Create a view named ‘Faculty_details’ that presents information about faculty members.



```
CREATE VIEW Faculty_details AS
SELECT Faculty_ID, First_Name ||' '|| Last_Name AS "Faculty Name", Email
FROM Faculty;
```

```
View created.
```

Fig 60: Command for Creating Simple View

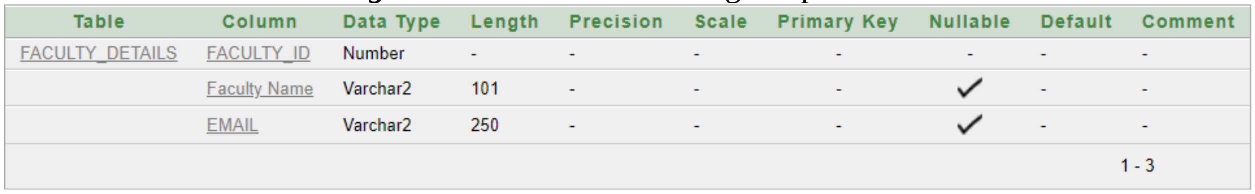
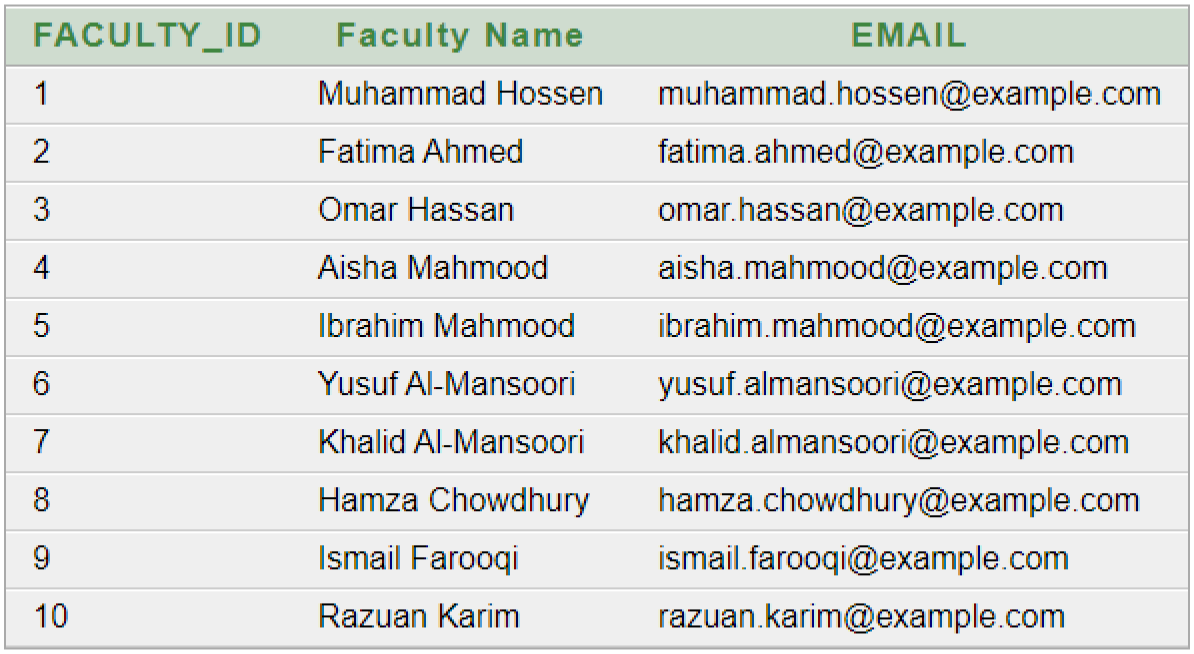


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FACULTY_DETAILS	FACULTY_ID	Number	-	-	-	-	-	-	-
	Faculty Name	Varchar2	101	-	-	-	✓	-	-
	EMAIL	Varchar2	250	-	-	-	✓	-	-

1 - 3

Fig 61: Description of the Simple view

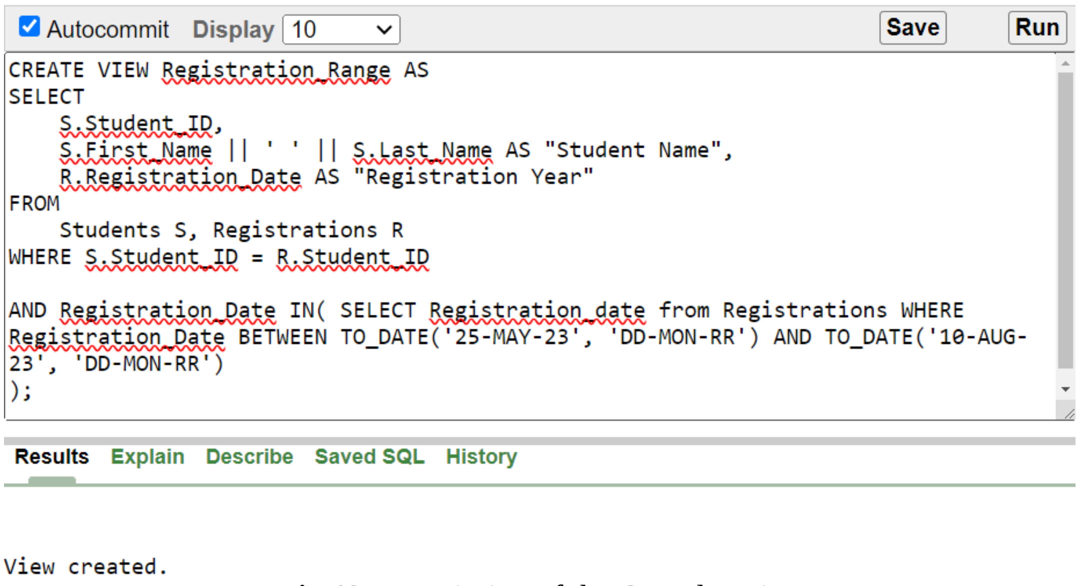


FACULTY_ID	Faculty Name	EMAIL
1	Muhammad Hossen	muhammad.hossen@example.com
2	Fatima Ahmed	fatima.ahmed@example.com
3	Omar Hassan	omar.hassan@example.com
4	Aisha Mahmood	aisha.mahmood@example.com
5	Ibrahim Mahmood	ibrahim.mahmood@example.com
6	Yusuf Al-Mansoori	yusuf.almansoori@example.com
7	Khalid Al-Mansoori	khalid.almansoori@example.com
8	Hamza Chowdhury	hamza.chowdhury@example.com
9	Ismail Farooqi	ismail.farooqi@example.com
10	Razuan Karim	razuan.karim@example.com

Fig 62: Result for Simple View

2. Complex view

Question 1: Create a view named 'Registration_range' that displays the student ID, student name, and registration year for students who registered between '25-may-23' and '10-AUG-23'. The view is based on the 'Students', 'Registrations' table.



```
CREATE VIEW Registration_Range AS
SELECT
    S.Student_ID,
    S.First_Name || ' ' || S.Last_Name AS "Student Name",
    R.Registration_Date AS "Registration Year"
FROM
    Students S, Registrations R
WHERE S.Student_ID = R.Student_ID

AND Registration_Date IN( SELECT Registration_date from Registrations WHERE
Registration_Date BETWEEN TO_DATE('25-MAY-23', 'DD-MON-RR') AND TO_DATE('10-AUG-
23', 'DD-MON-RR')
);
```

```
View created.
```

Fig 63: Description of the Complex view

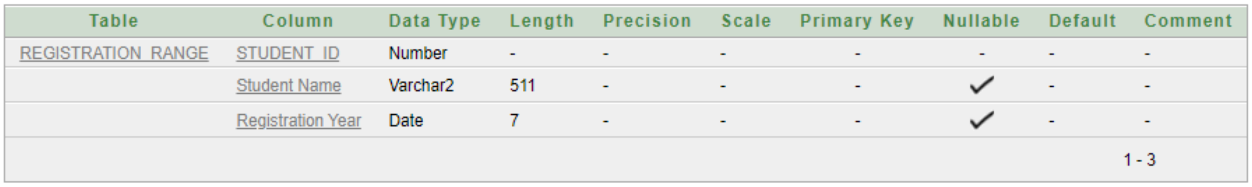
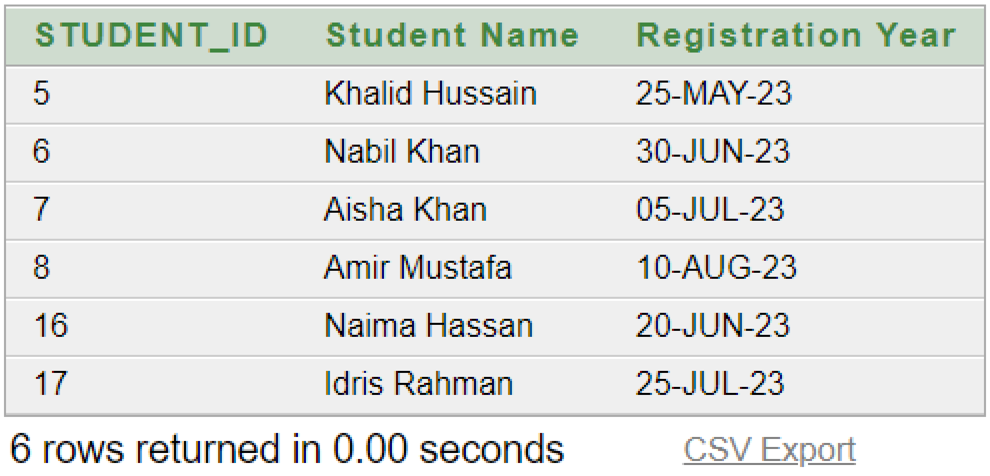


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REGISTRATION_RANGE	STUDENT_ID	Number	-	-	-	-	-	-	-
	Student Name	Varchar2	511	-	-	-	✓	-	-
	Registration Year	Date	7	-	-	-	✓	-	-

1 - 3

Fig 64: Command for Creating Complex View



STUDENT_ID	Student Name	Registration Year
5	Khalid Hussain	25-MAY-23
6	Nabil Khan	30-JUN-23
7	Aisha Khan	05-JUL-23
8	Amir Mustafa	10-AUG-23
16	Naima Hassan	20-JUN-23
17	Idris Rahman	25-JUL-23

6 rows returned in 0.00 seconds   CSV Export

Fig 62 : Result of Complex View

Conclusion

In summary, the course management system project has been successfully implemented, providing a robust platform for managing courses, students, faculties, registrations and administrative tasks efficiently. Through the utilization of SQL, we've established a structured database schema that facilitates seamless data organization and retrieval and also there are several potential future aspects to enhance and expand upon this project such as the user interface to make it more intuitive and user-friendly and Implement data analytics capabilities to provide insights into course performance, student progress. Consider integrating features such as online course registraions , grade tracking, and communication tools to further enhance the functionality of the system. By focusing on these future aspects, we can continue to evolve and improve the course management system, ultimately providing a more comprehensive and efficient solution for educational institutions and users alike.