DATABASE MANAGEMENT SYSTEMS (10211CS207)

TASK:12 MICRO PROJECT

|  |  |  |
| --- | --- | --- |
| **TeamDetails :** |  |  |
|  |  |  |
|  |  |  |
| **Team Members:** |  |  |
| **NAMES:** | **VTU:** | **Registration no:** |
| M.Neha | (VTU29338) | (24UECS0754) |
| K.Nikhitha | (VTU29298) | (24UECS0639) |
| B.Mokshitha | (VTU29865) | (24UECS1143) |
| K.Renu sri | (VTU27665) | (24UECS0681) |
| **Under the guidance of:**  Dr.N.K.Senthil Kumar  professor |  |  |

TITLE

STUDENT RESULT MANAGEMENT SYSTEM

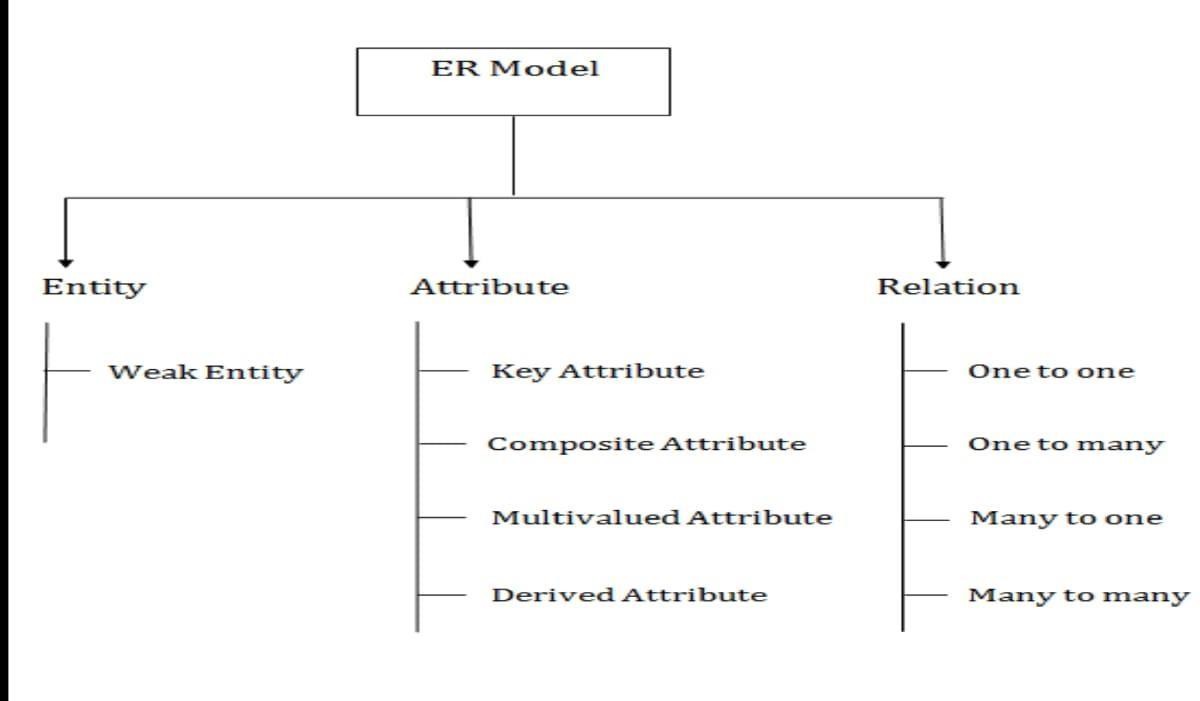
1.ER Diagram:

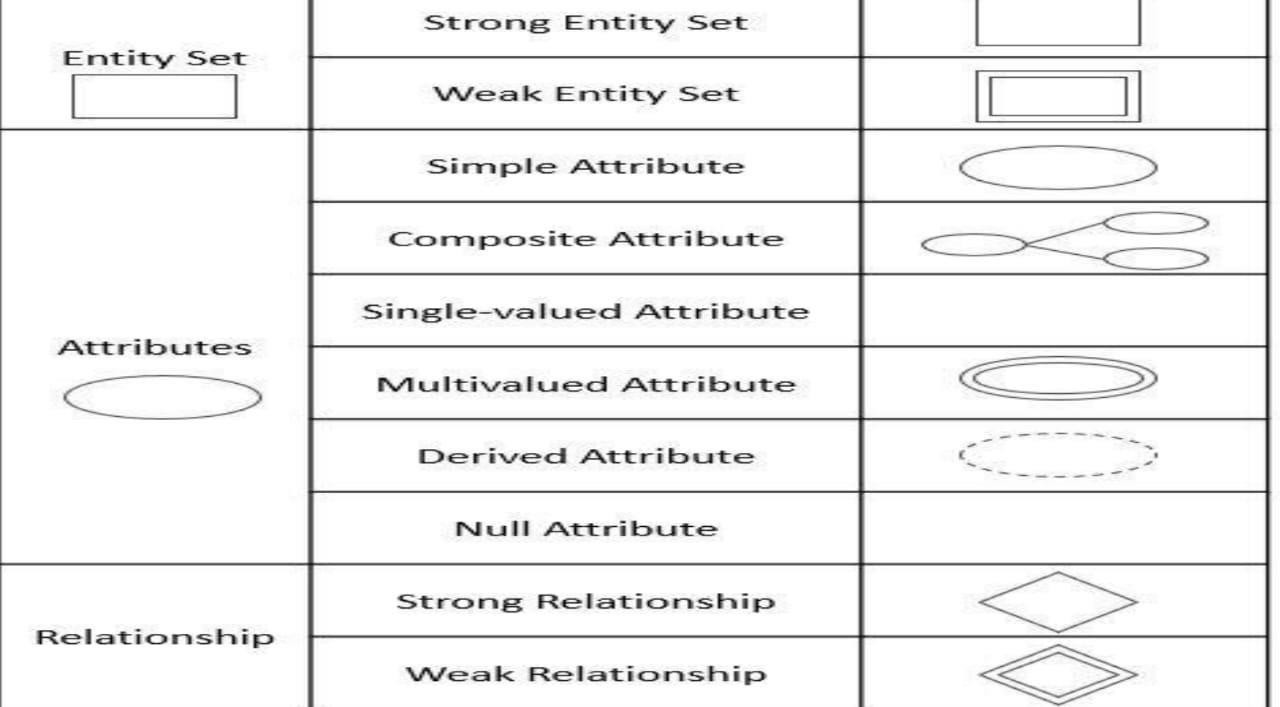
Aim : To draw the conceptual design for student result management system

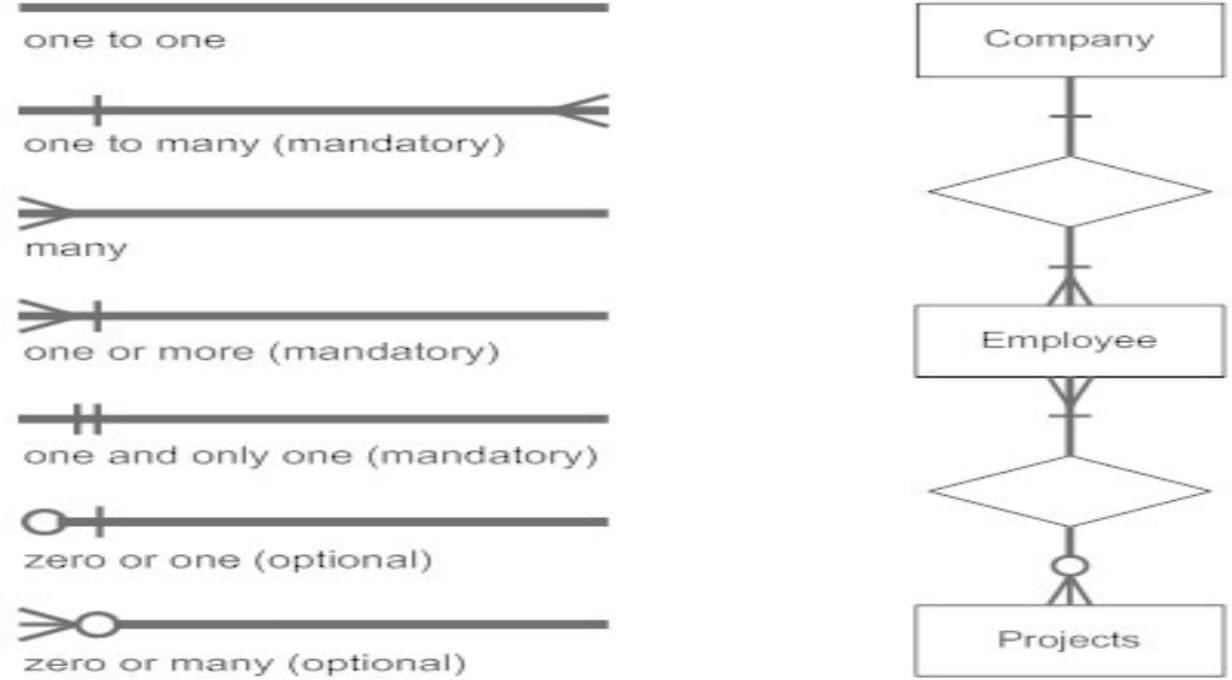
### E-R Diagram

Entity–Relationship model:

* ER model stands for an Entity-Relationship model. It is a highlevel data model. This model is used to define the data elements and relationship for a specified system.
* It develops a conceptual design for the database. It also develops a very simple and easy to design view of data
* It develops a conceptual design for the database. It also develops a very simple and easy to design view of data







**WEAK ENTITY:** An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

**ATTRIBUTE:** The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute

**KEY ATTRIBUTE:**The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.

**COMPOSITE ATTRIBUTE**:An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

**MULTI VALUED ATTRIBUTE :**An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

**DERIVED ATTRIBUTE:** Attributes which are derived from other attributes

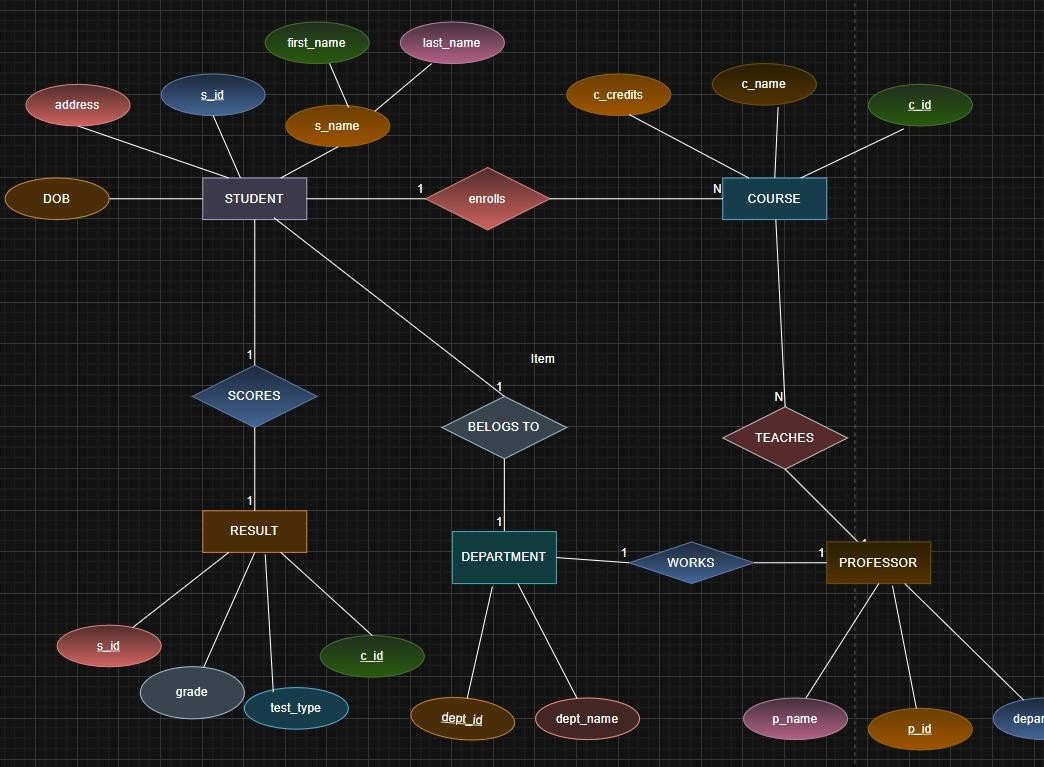
ER-MODEL FOR STUDENT RESULT MANAGEMENT SYSTEM

**Entities**: Student, Course, Professor, Department,Result

### Attributes:

* Student: S\_ID,s\_name, First\_Name, Last\_Name, DOB,Address
* Course: C\_ID, C\_Name,C\_Credits
* Professor: P\_ID, P\_Name, Department
* Department: DepartmentID, Department
* Result:s\_id,grade,test\_type,c\_id

### Relations:

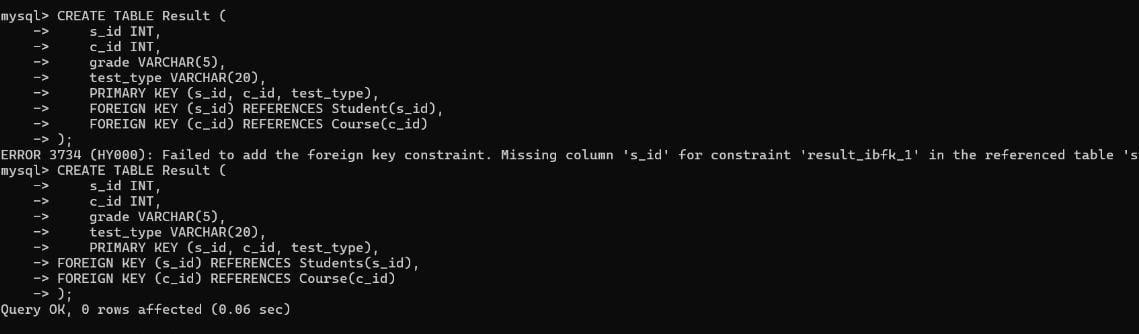
1. Enrolls In relation (between Student and Course):
   * Foreign Key: Student ID in the Course entity
   * Foreign Key: CourseID in the Student entity
2. Teaches relation (between Professor and Course):
   * Foreign Key: Professor ID in the Course entity
   * Foreign Key: CourseID in the Professor entity
3. Belongs To relation (between Student and Department):
   * Foreign Key: StudentID in the Result entity
   * Foreign Key: Result\_ID in the Student entity
4. Works In relation (between Professor and Department):
   * Foreign Key: Professor\_ID in the Result entity
   * Foreign Key:Result\_ID in the Profressor entity

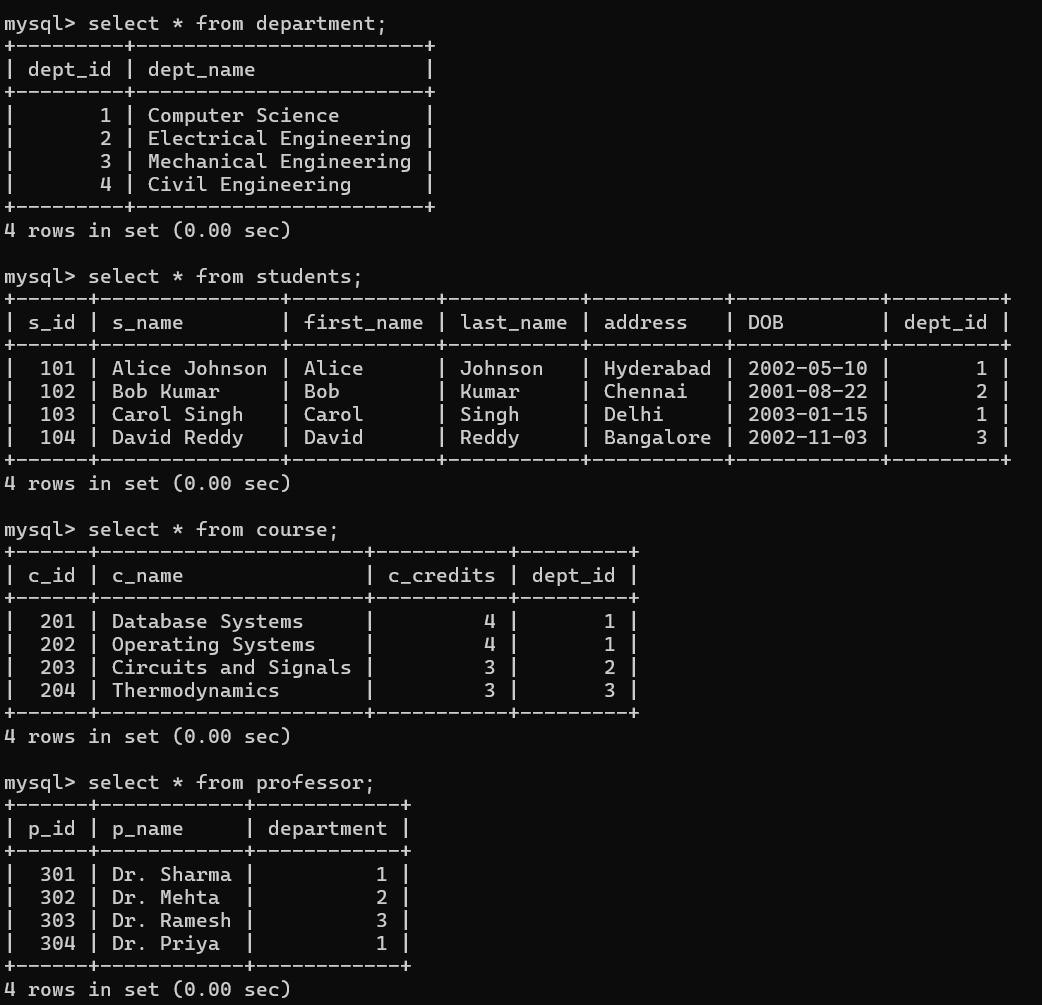
**Result:** Thus, the creating er diagram is completed successfully.

**Aim:** To execute relational operations, SQL aggregates, Joint queries for library management system.

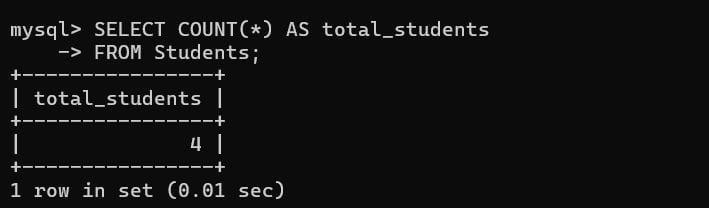
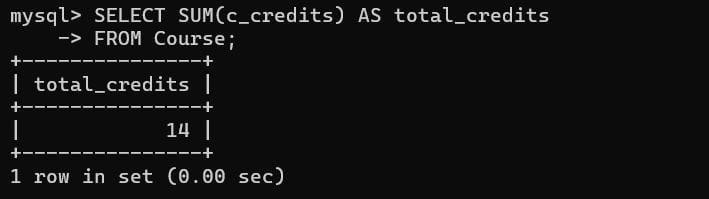
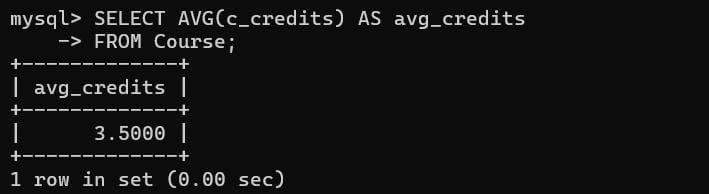
**Relational operations:**

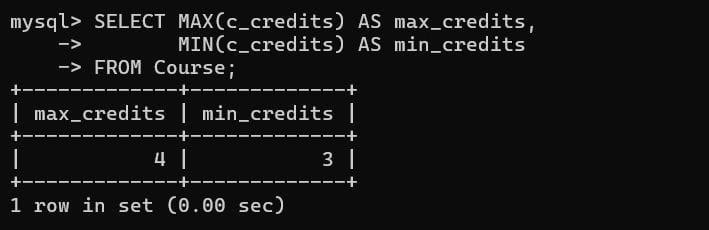
* + - **Creating and inserting values into the table**

1. **Project operation and union operation:**
2. **Select operation:**

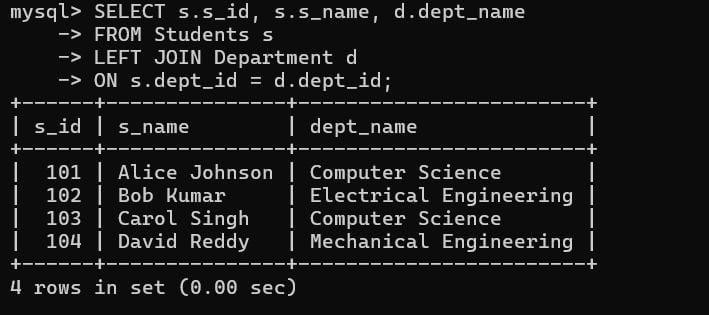
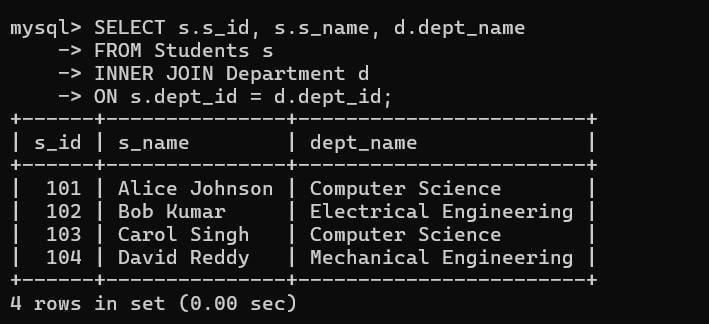
****

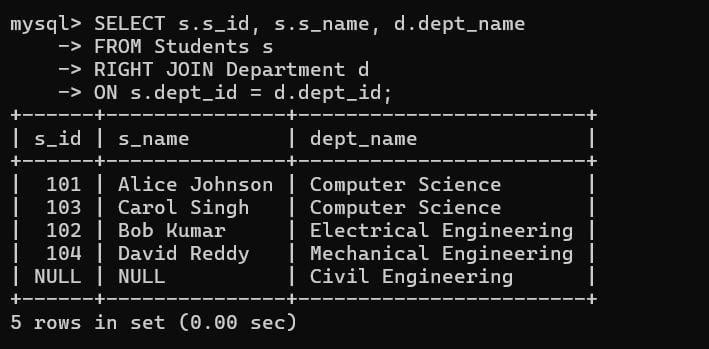
**SQL Aggregate**

1. **Count:**
2. **Sum:**
3. **Average:**
4. **Maximum and Minimum**

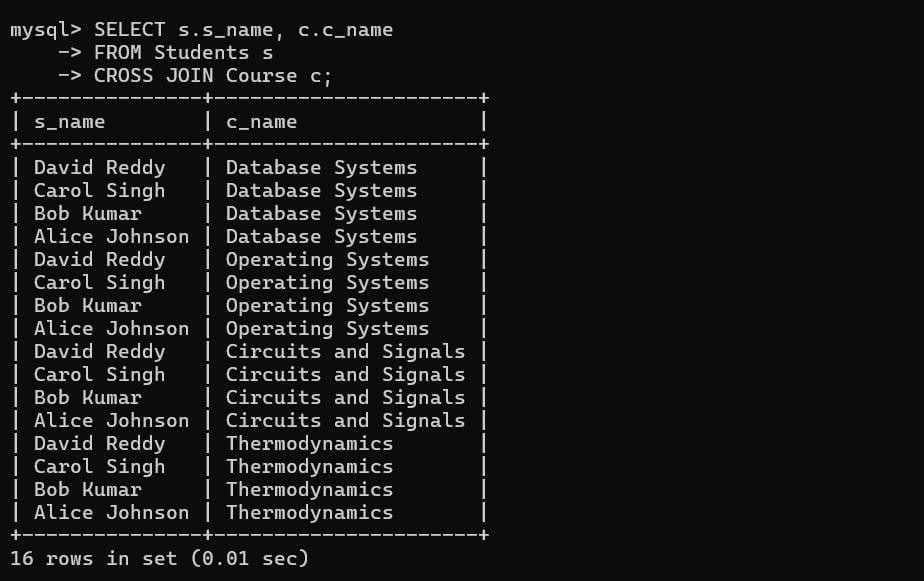
**:**

# JOIN QUERIES

* 1. **LEFT JOIN**
  2. **INNER JOIN**
  3. **RIGHT JOIN**

****

* 1. **CROSS JOIN**

****

RESULT**:** Thus, to execute relational operations, SQL aggregates, join queries for library management system is successfully executed.

Normalization

Normalization in the context of databases refers to the process of organizing data in a database efficiently. The goal is to reduce data redundancy and dependency by organizing fields and table of a database. This helps in minimizing the anomalies that can arise when modifying the data.

There are several normal forms (NF) that define the levels of normalization, with each normal form addressing different types of issues:

## First Normal Form (1NF):

* + - Eliminate duplicate columns from the same table.
    - Create a separate table for each group of related data and identify each row with a unique column or set of columns.

## Second Normal Form (2NF):

* + - Meet all the requirements of 1NF.
    - Remove partial dependencies—ensure that non-prime attributes are fully functionally dependent on the primary key.

## Boyce-Codd Normal Form (BCNF):

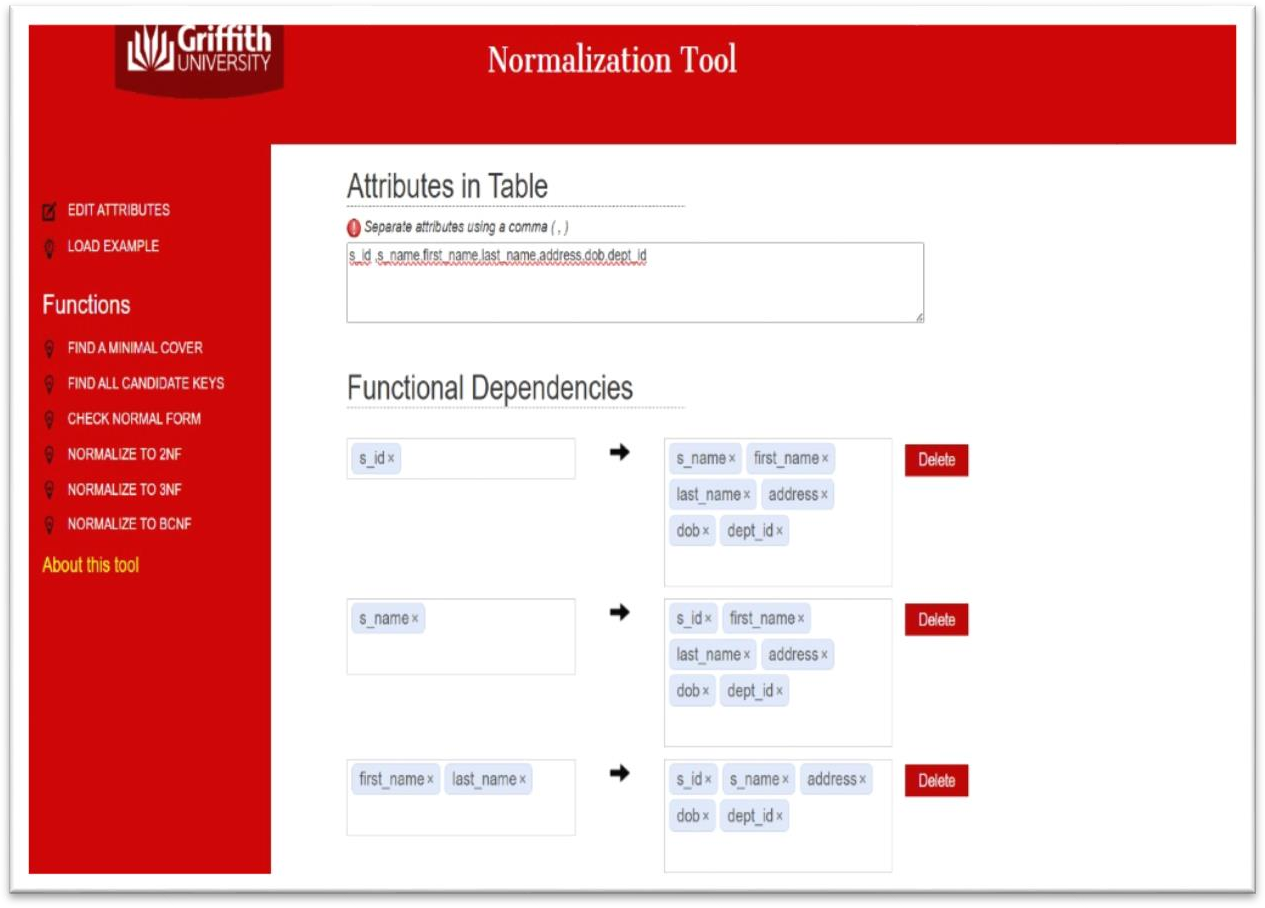
* + - * A more stringent form of 3NF.
      * For a table to be in BCNF, it must satisfy an additional requirement compared to 3NF, dealing specifically with certain types of functional dependencies.
* In this database we perform normalisation using Griffith university normalisation tool

Steps to follow for doing normalisation using Griffith normalisation process:

**Step1:** search for ***Griffith university normalisation*** *tool* in web browser

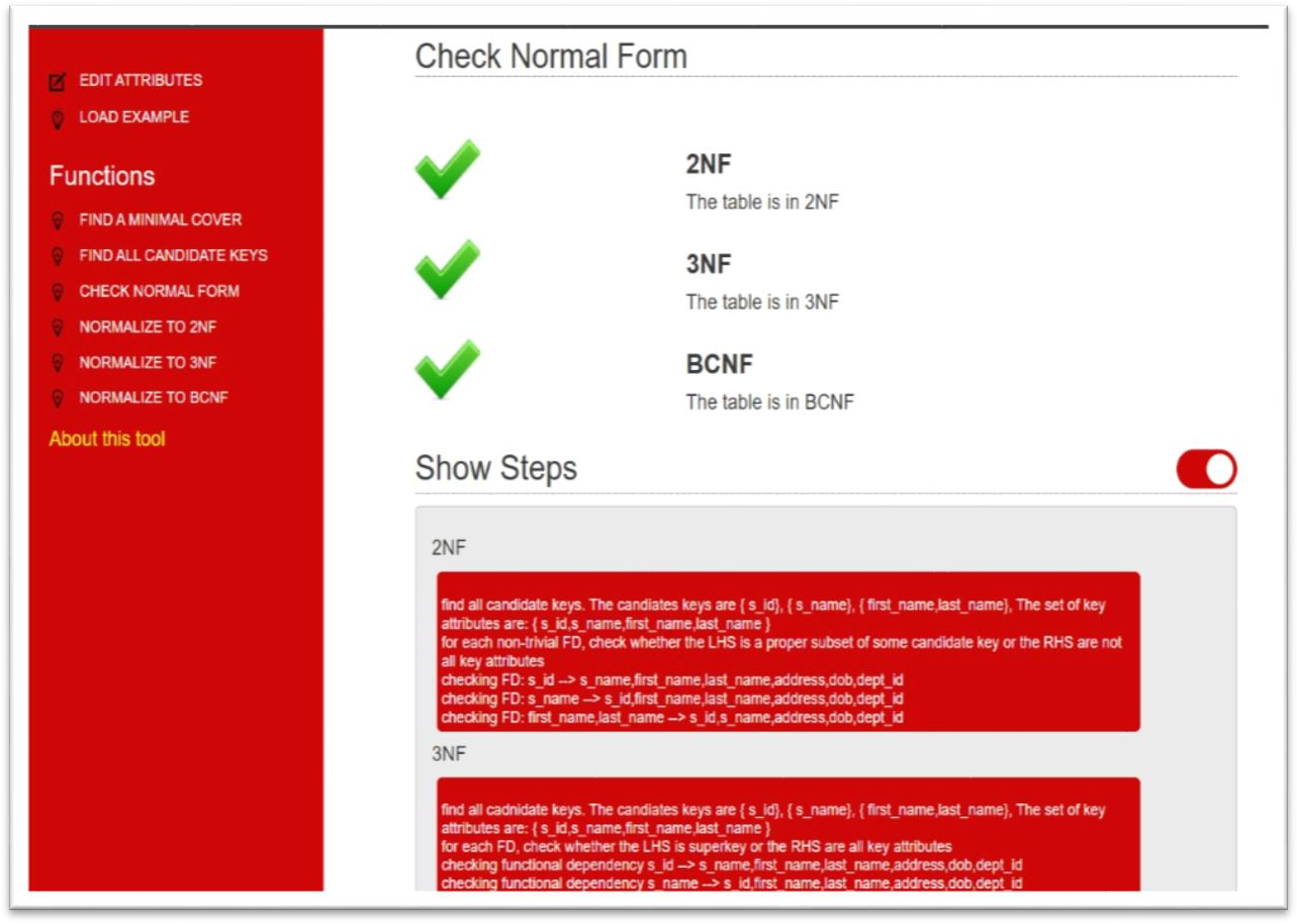
**Step2**: After opening the tool enter the attributes of the entity S\_ID, s- name, address, dob, s-course. Make sure to separate the attributes using commas in between them.

**Step3**: Add the dependencies of the attributes



Do as per your entity and add the dependencies as shown in the fig.

Step4: In the left if the window below functions on **checknormalform**option



We will get the screen shown above the normal form of the given attributes is checked (BCNF)

And the following steps are displayed below:

# 2NF:

1. find all candidate keys. The candiates keys are { s\_id}, { s\_name}, { first\_name,last\_name}, The set of key attributes are: { s\_id,s\_name,first\_name,last\_name }
2. for each non-trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes
3. checking FD: s\_id --> s\_name,first\_name,last\_name,address,dob,dept\_id
4. checking FD: s\_name --> s\_id,first\_name,last\_name,address,dob,dept\_id
5. checking FD: first\_name,last\_name --> s\_id,s\_name,address,dob,dept\_id.

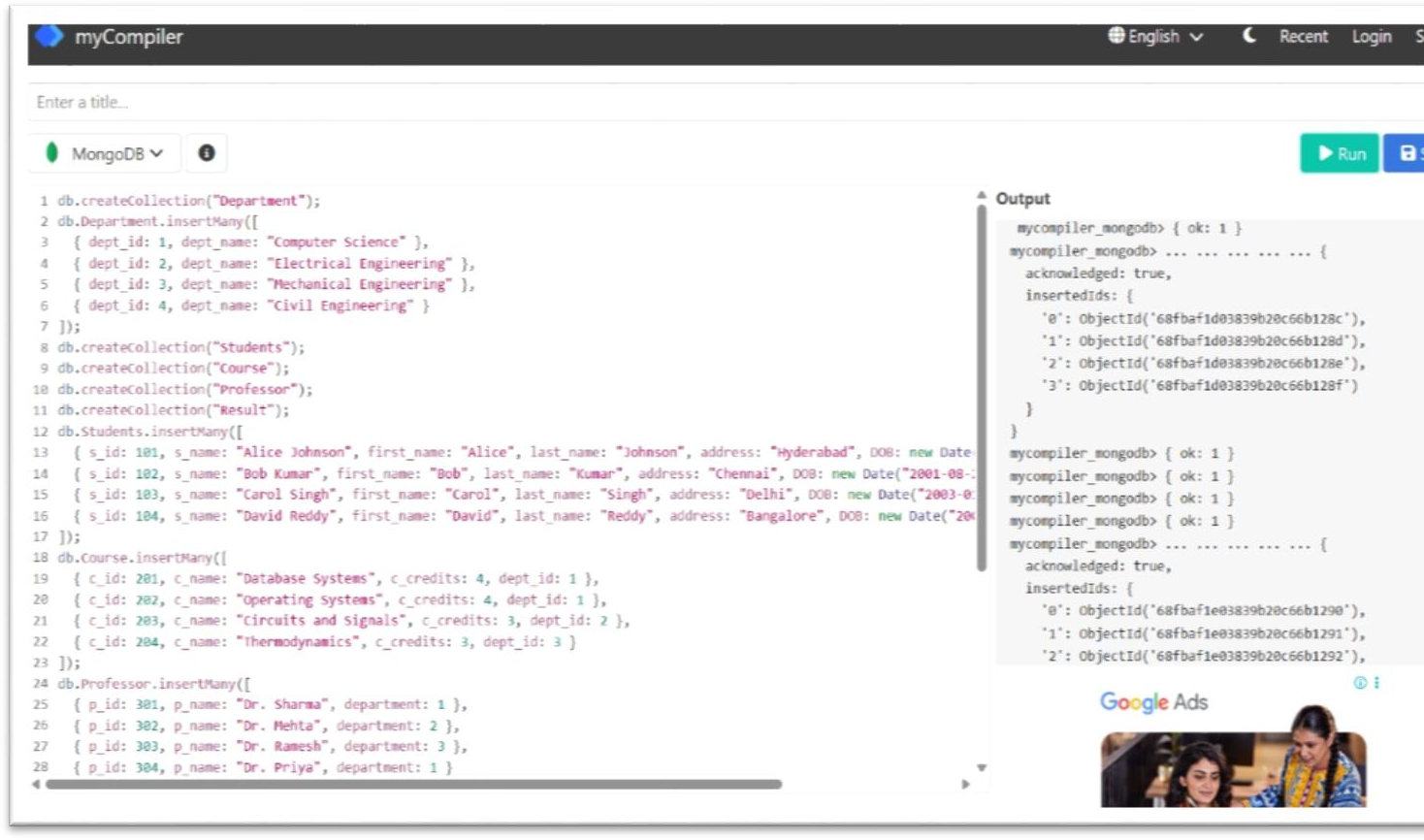
# 3NF:

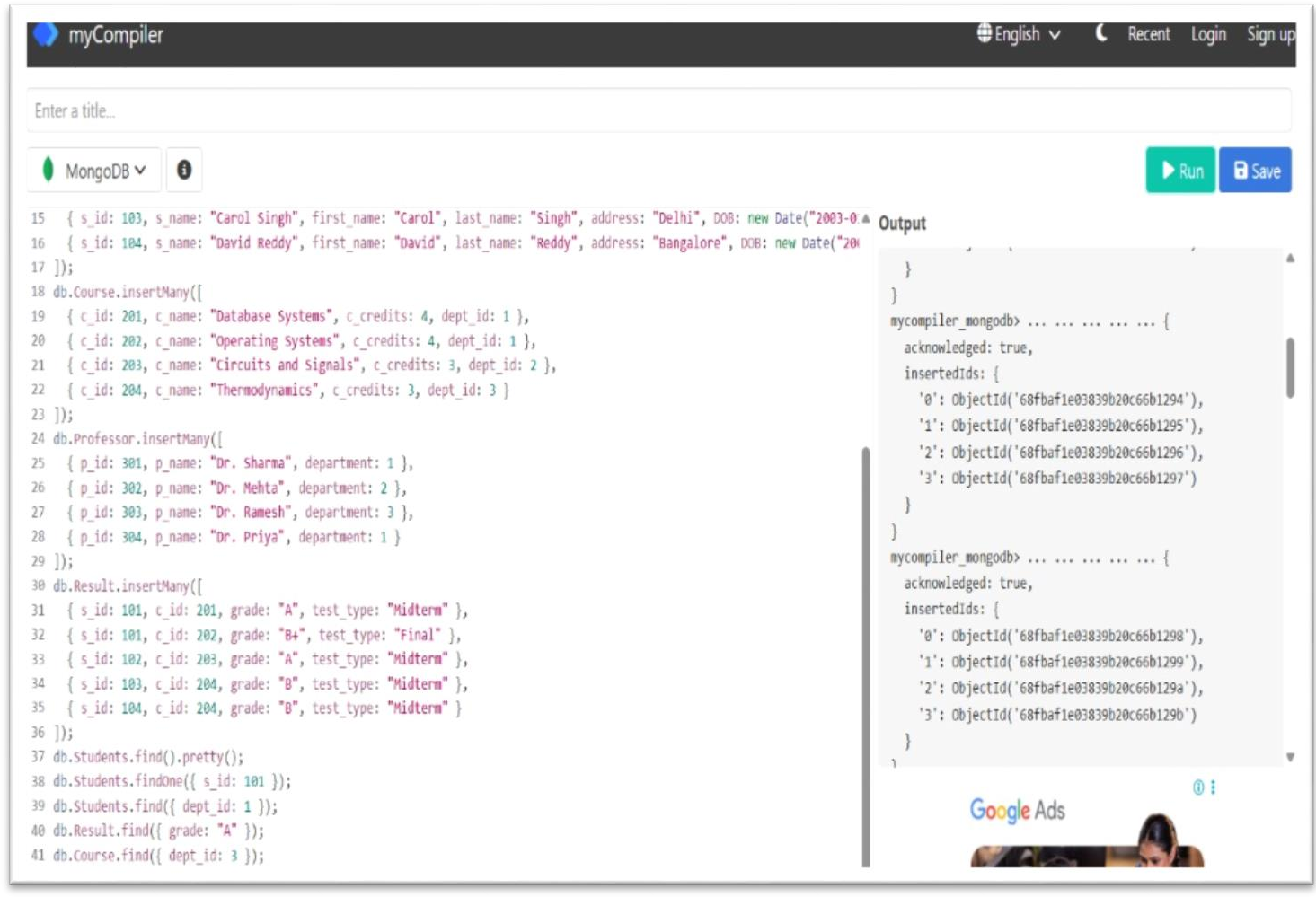
1. find all cadnidate keys. The candiates keys are { s\_id}, { s\_name}, { first\_name,last\_name}, The set of key attributes are: { s\_id,s\_name,first\_name,last\_name }
2. for each FD, check whether the LHS is superkey or the RHS are all key attributes
3. checking functional dependency s\_id --> s\_name,first\_name,last\_name,address,dob,dept\_id
4. checking functional dependency s\_name --> s\_id,first\_name,last\_name,address,dob,dept\_id
5. checking functional dependency first\_name,last\_name --> s\_id,s\_name,address,dob,dept\_id

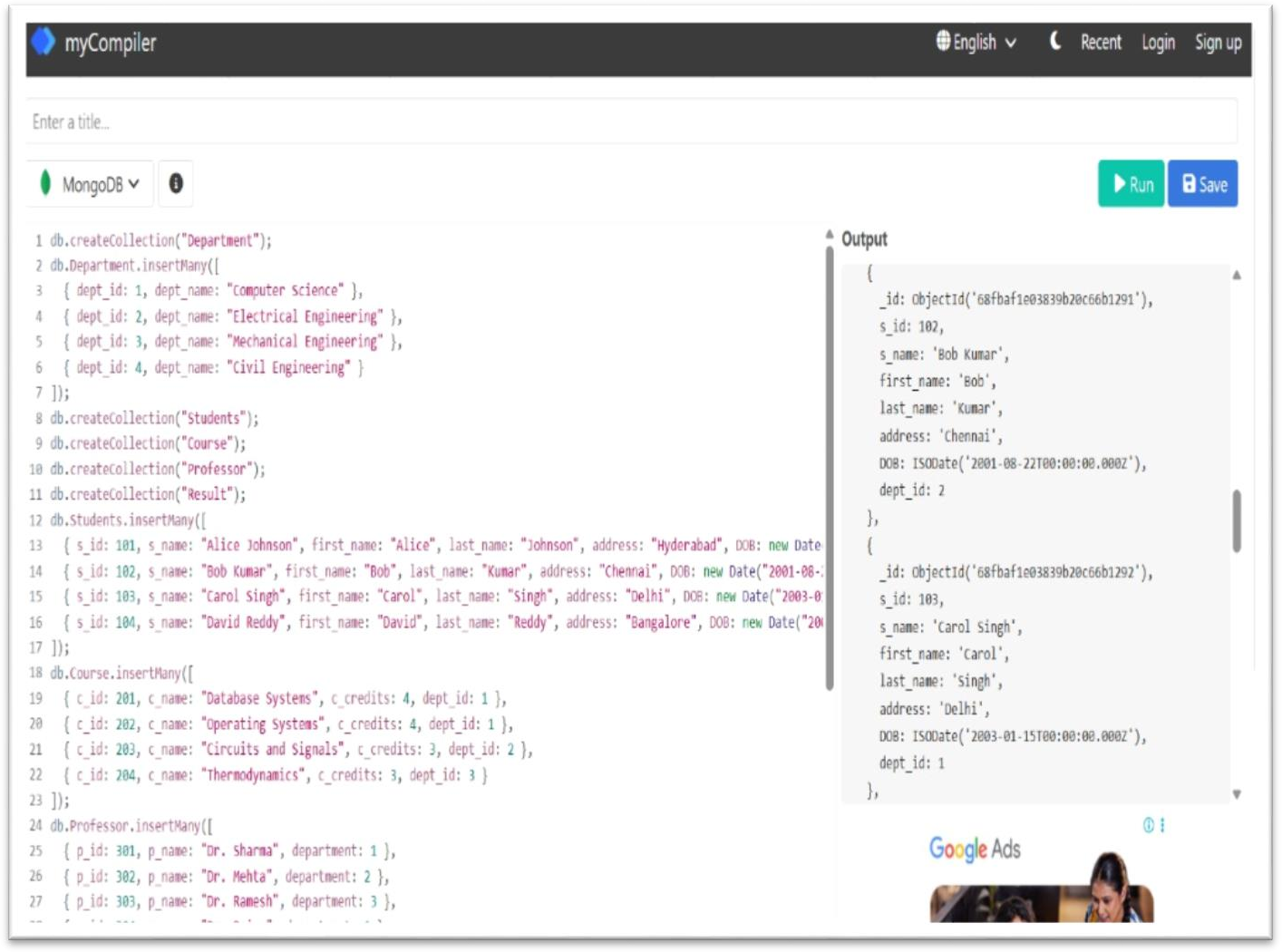
# BCNF:

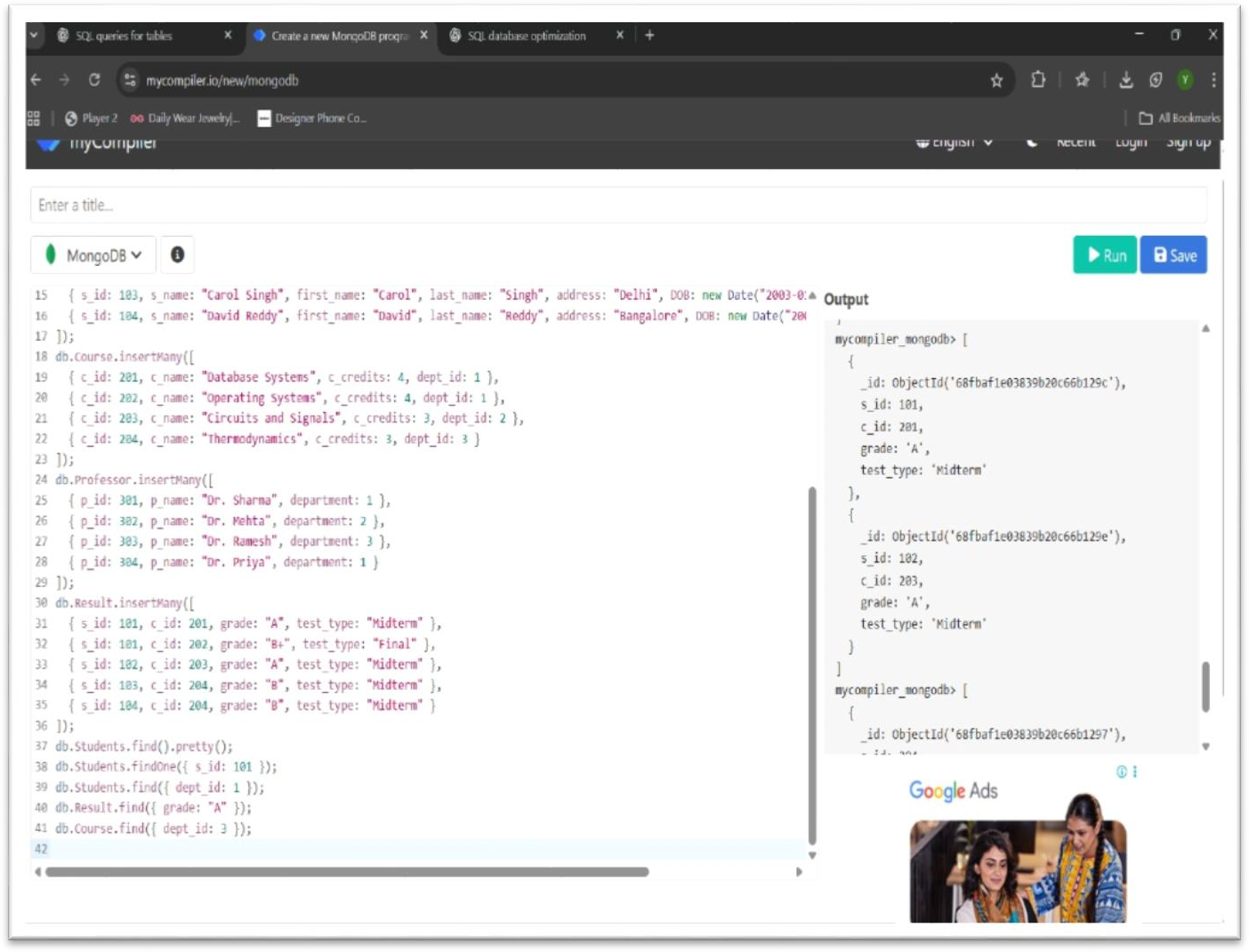
1.A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

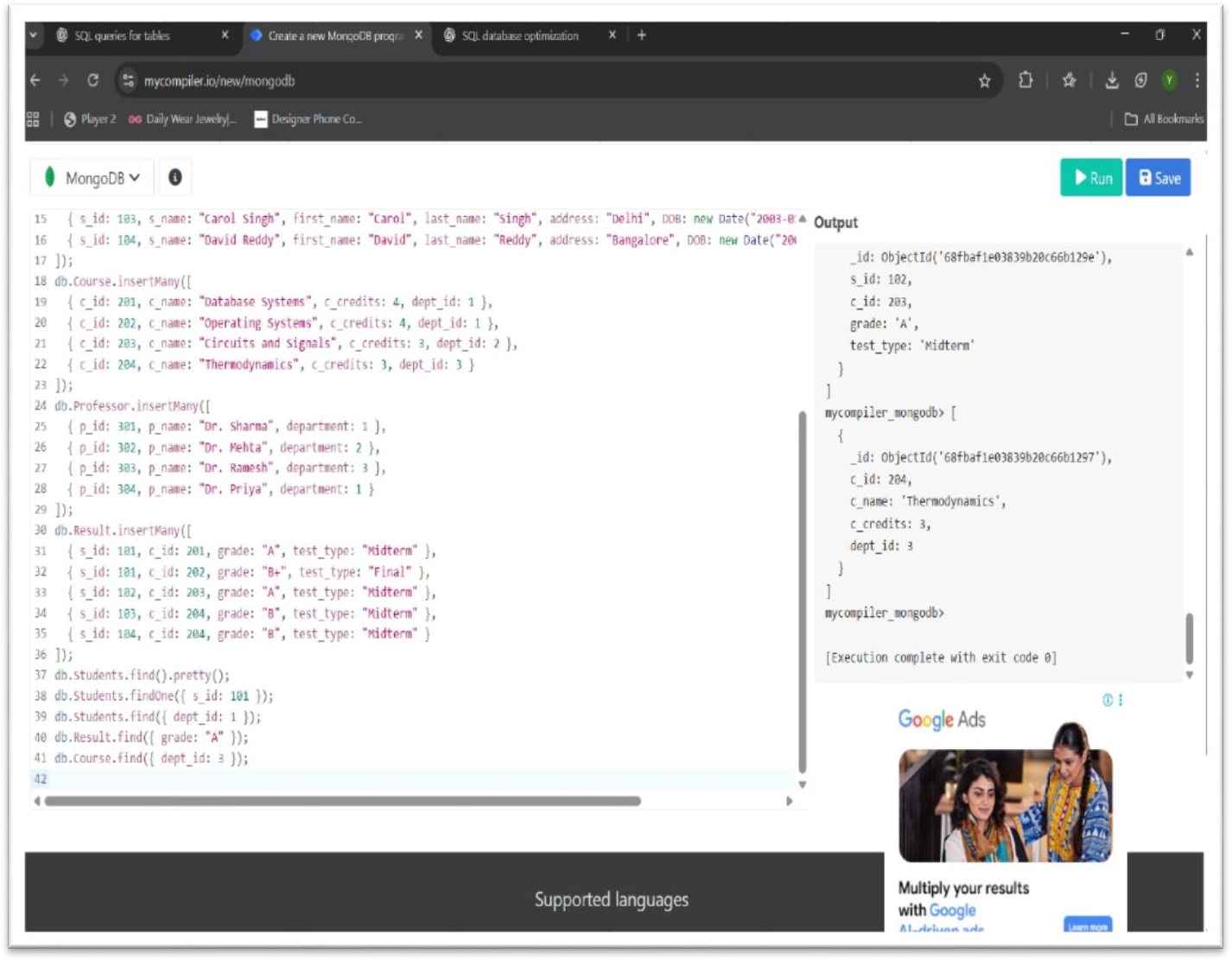
**Result:** Thus, the normalization to 1nf,2nf,3nf, BCNF is completed successfully.

**Aim:** To implement the document database and graph database by using Mon gosh.









**Result**: Thus implemented the document database and graph database by using Mon gosh.