**C-406** DAY-1

**Foundations of SRE -RDBMS and SQL**

**Priyansh Kakani** 10-02-2025

* Today I have learned about the SQL queries and implemented through the MySql.
* Got a different view about table creation through a cupboard example which was given by the trainer. helped to identify how the SQL works in searching.
* To create a DATABASE and use it
  + CREATE DATABASE SCHOOL;
    - This creates a database SCHOOL.
  + USE SCHOOL;
    - This uses the specific database SCHOOL.
* To create a table under the DATABASE SCHOOL

CREATE TABLE STUDENT (

ID INT,

NAME VARCHAR2(20),

STREAM VARCHAR2(20),

PRIMARY KEY(ID)

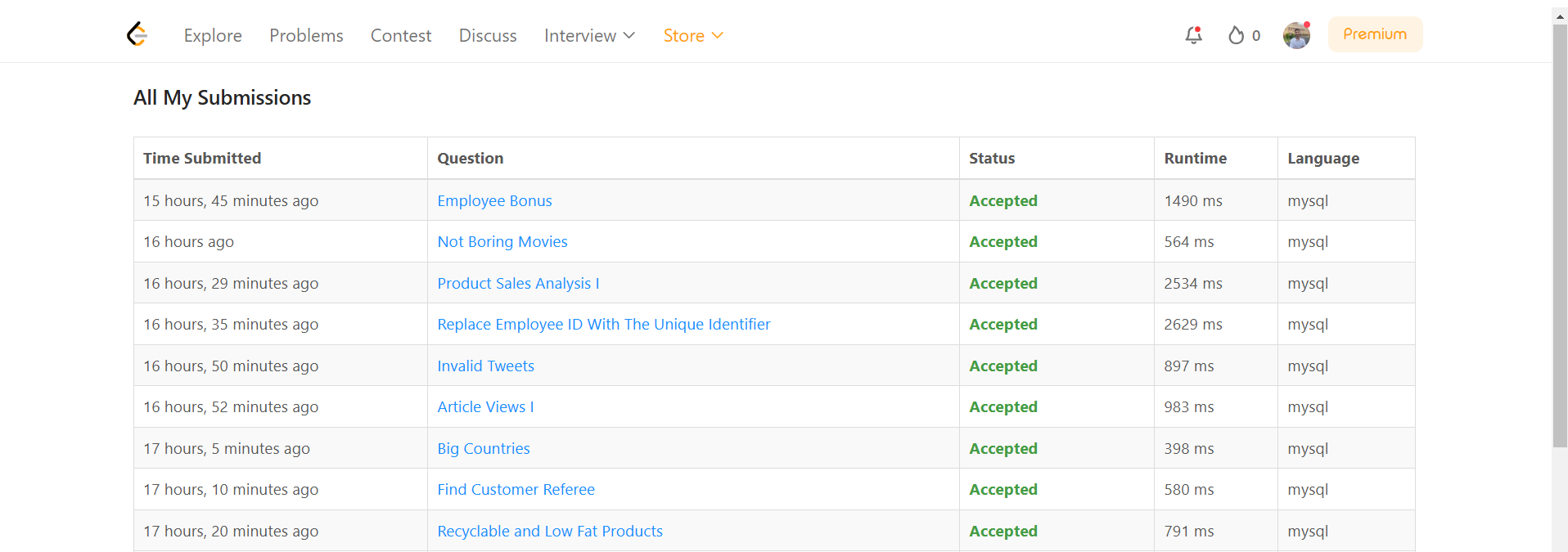
);

* In most of the cases DDL commands are faster than the DML commands
  + For e.g. TRUNCATE is faster than the DELETE just takes 0.000 sec to be executed.
* Used RENAME for renaming the table by 2 ways.
  + RENAME TABLE STUDENT TO NEW\_STUDENT;
  + ALTER TABLE STUDENT RENAME TO NEW\_STUDENT;
* Learned how the deep copies and shallow copies are created in the SQL.
  + 2 Ways for copying the old student relation data to new student relation.
  + For the Deep copy we use
    - CREATE TABLE OLD\_STUDENT SELECT \* FROM OLD\_STUDENT;
  + For the Shallow copy we use
    - CREATE TABLE NEW\_STUDENT LIKE OLD\_STUDENT;
      * It just creates the schema only no data is copied.
* To get a view of the table as per requirements we use SELECT.
  + SELECT \* FROM STUDENT;
    - This query will get all the columns of the STUDENT table.
* By default if we do sorting of table rows
  + it is ASCENDING ALWAYS just mention ORDER BY COLUMN\_NAME.
  + to get descending order use ORDER BY COLUMN\_NAME DESC.
* To get the description about the table e.g. columns, datatypes.
  + DESC STUDENT;
* To create a temporary table
  + CREATE TEMPORARY TABLE TEMP\_STUDENT(

… COLUMN AND DATATYPE DECLARATION

)

* Whenever we use HAVING It is mandatory to use GROUP BY otherwise it will not work. But when using GROUP BY it is optional to use HAVING.
* The primary key will be the first default key while retrieving the data from the table
* The data is aggregated by how it is spread across on the memory disk .if we specify the GROUP BY . then it will be of specific order.
* When we use GROUP BY so there are multiple table are created with the UNIQUE COLUMN values and have multiple related rows. And after that result are filtered out.
* Using the HAVING with GROUP BY is just like using the WHERE with FROM for the filter of the results.
* Used the wildcard for the searching patterns
  + e.g. SELECT NAME FROM STUDENT LIKE ‘P%’
    - it will print the student name starts with the P.
* Learned the aggregation functions such as COUNT,AVG,MIN,MAX with the GROUP BY and having
* Done today 9 SQL queries today on leetcode basics.



**C-406** DAY-2

**Foundations of SRE -RDBMS and SQL**

**Priyansh Kakani** 11-02-2025

* Today we have learned how to set the permission to the users for using the database as the different access is given to the different users so we provide a number that is decimal equivalent by the formula 2^n where n is the 1 present in the binary representation of the number.
* Sample program to demonstrate how to create permission table and insert data into it in decimal format.
  + CREATE TABLE permissions (  
        user\_id INT PRIMARY KEY,  
        username VARCHAR(50),  
        permission\_flags INT  -- Will store permission bits  
    );  
      
    -- Insert sample data  
    INSERT INTO permissions (user\_id, username, permission\_flags) VALUES  
    (1, 'admin', 7),     -- Binary: 111 (Read: 1, Write: 1, Execute: 1)  
    (2, 'developer', 6), -- Binary: 110 (Read: 1, Write: 1, Execute: 0)  
    (3, 'viewer', 4),    -- Binary: 100 (Read: 1, Write: 0, Execute: 0)  
    (4, 'guest', 1);     -- Binary: 001 (Read: 0, Write: 0, Execute: 1)
  + To give permission to user about execute we can change the values as
    - update permissions   
      set permission\_flags =permission\_flags ^ 1  
      where (permission\_flags & 1)=0;
* to give permission to user about write we can set values by
  + update permissions  
    set permission\_flags = permission\_flags | 2  
    where (permission\_flags & 2) =0
* to give permission to user about read we can set values by
  + select   
    username,  
    permission\_flags & 4 as has\_read\_permission,  
    case  
       when permission\_flags & 4 > 0 then 'Yes'  
       else 'No'  
    end as can\_read  
    from permissions;
* SET SQL\_SAFE\_UPDATES=0.
  + The error occurs because MySQL's **safe update mode** requires a key column in the WHERE clause to prevent accidental updates. Without it, MySQL blocks the update to ensure data safety.
* We can also do manipulation using bits and it applications are also the multiplication and division.
  + If we shift 1 bit to right then the number will be divided by 1.
  + If we shift 1 bit to left then the number will be multiplied by 2.
* Bit shifting operations example  
  CREATE TABLE bit\_shift\_demo (  
      id INT PRIMARY KEY,  
      value INT  
  );  
  INSERT INTO bit\_shift\_demo (id, value) VALUES  
  (1, 8),  -- Binary: 1000  
  (2, 12),  -- Binary: 1100  
  (3, 16);  -- Binary: 10000
* example to left shift the values will be double and 4 times

select id,value,  
value << 1 as left\_shft\_1,  
value << 2 as left\_shift\_2  
from bit\_shift\_demo;

* UNION operator is used to combine the results of two or more SELECT queries into a single result set. It removes duplicate rows from the result set by default.
  + SELECT column1, column2, ... FROM table1 WHERE condition1 UNION SELECT column1, column2, ... FROM table2 WHERE condition2;
* In UNION ALL it contains all rows even they are same i.e. redundant rows will be there in result.
* UNION ALL is faster than the UNION as it doesn’t checks for the repeated data into the result.
* SELECT 1 FROM CUSTOMERS;
  + This query selects 1 if there is a valid row so if we are applying into customers if customers has 5 rows so it will print 5 1’s in column each 1 denoting to each line used to check the no of rows present in table.
  + In practical scenarios if we want to count the no of rows indirectly then we can use this statement with where clause and a aggregate function e.g. COUNT so it will count no of rows.
* ANY keyword in SQL is typically used with a comparison operator (like =, >, <, etc.) to compare a value to any value in a set of values returned by a subquery. It allows you to compare a value with multiple values, and if any of those values meet the condition, the result is true.
  + SELECT column\_name FROM table\_name

WHERE column\_name operator ANY (subquery);

* + E.g. SELECT sale\_id, amount, salesperson FROM sales

WHERE amount = ANY (SELECT amount FROM sales WHERE salesperson = 'Bob');

This will select all amount if the salesperson is Bob.

* Sometime we use left join which is less expensive in terms of time than the inner join.
  + E.g. when creating a same column in the table representing the manager ‘s manager in the same table w/o using the INNER JOIN.
* DENSE\_RANK is a window function that assigns a unique rank to each distinct value in a result set, without leaving gaps in the rank values when there are ties.
* The PARTITION BY keyword in SQL is used in window functions to divide the result set into partitions (or groups) before performing the function.
  + E.g. we want the sales comparision between the salespersons and their monthly sales.
* the OVER keyword is used to define the windowing for the function, specifying how the data should be ordered or partitioned.
* RANK function is not good when there are 2 values same then it will give same rank to all 3 of them and continue to next with previous rank+2 to next which is not accurate when it comes to ranking. Example-

| **order\_id** | **sale\_amount** | **rank** |
| --- | --- | --- |
| 1 | 300 | 1 |
| 2 | 200 | 2 |
| 3 | 200 | 2 |
| 4 | 150 | 4 |
| 5 | 100 | 5 |

* LAG is a window function in SQL that allows you to access data from a previous row in the result set without the need for self-joins.
  + Its syntax is LAG(expression, offset, default\_value) OVER (PARTITION BY partition\_column ORDER BY order\_column).
  + E.g. if amazon has 5 products and we want to compare the products revenue according to the month. Then the LAG and DENSE\_RANK is used.

**C-406** DAY-3

**Foundations of SRE -RDBMS and SQL**

**Priyansh Kakani** 12-02-2025

* today we have learned about the primary keys and foreign keys how they react
* Declarations of primary key are:
* ID INT PRIMARY KEY
* ID INT, PRIMARY KEY(ID)
* Declarations of foreign key are:
  + CREATE TABLE table\_name (

column\_name datatype,

column\_name datatype,

...CONSTRAINT fk\_name

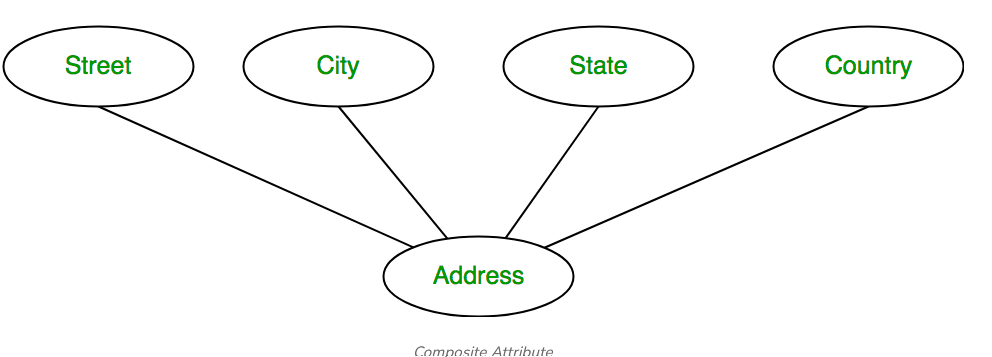
FOREIGN KEY (column\_name) REFERENCES

other\_table (primary\_key\_column));

* + If we insert a data in table 2 which has contain the foreign key as the primary key from the table 1 if the value of foreign key doesn’t matches in the table 1 then it will throw an error.
* For e.g. I have courses table courseID as primary key and students table has studentID has primary key and table enrollment which has EnrollmentID and has foreign key as courseID and EnrollmentID so if we push a result into enrollment table with invalid courseID not present in the courses table so it will throw an error.
* The main work of the foreign key is to remove the data duplication and also it does referencing the other table as the cherry on the top.
* An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.
* ATTRIBUTES are the properties that define the entity type and its types
  + This is a key attribute also can be said as primary key representation.



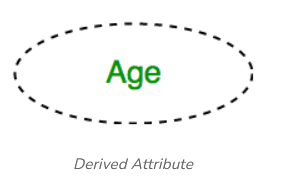
* + Composite attribute that consist of multiple attribute



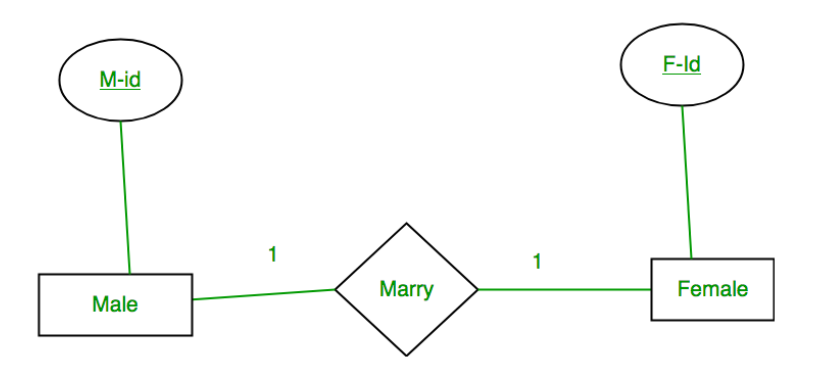
* + Multivalued attribute that contains multiple values as a person can have 2 phone numbers



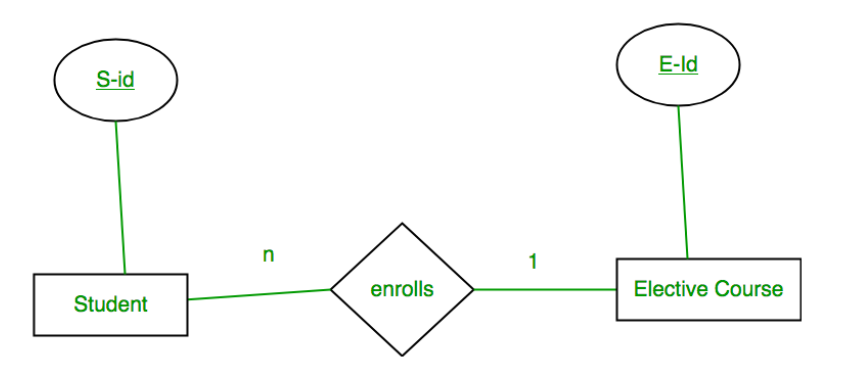
* + An attribute that can be derived from other attributes of the entity type is known as a derived attribute



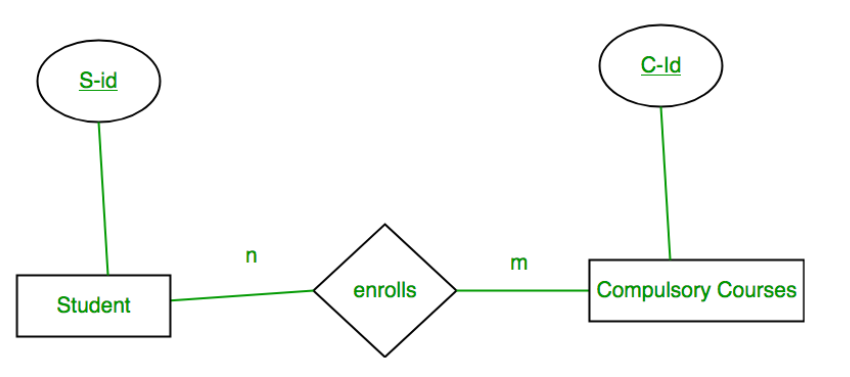
* An **Entity-Relationship (ER) Model** is a conceptual framework used to design and represent the structure of a database. It defines entities (objects) and the relationships between them, using diagrams with symbols like rectangles for entities and diamonds for relationships.
  + 1:1 mapping as only one male can marry one women only and vice versa.



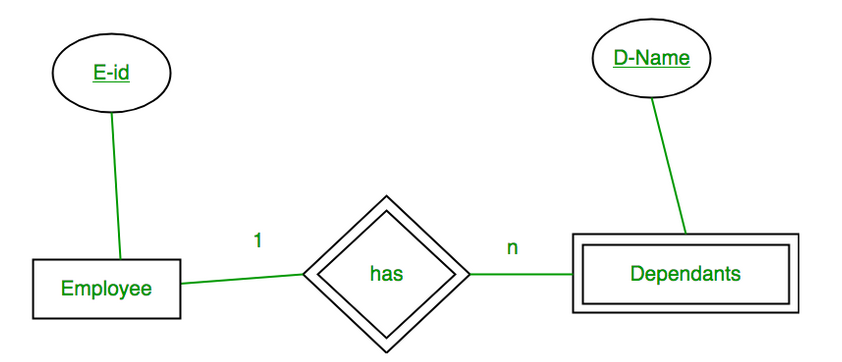
* + N:1 mapping as n entities can be mapped with single entity e.g. n number of students can take a elective course but a multiple elective can’t be taken by a same student.



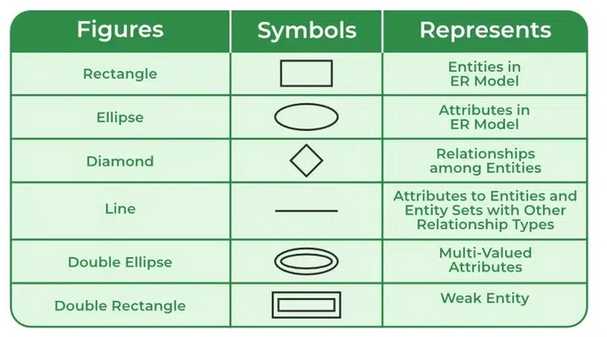
* + N:M mapping where multiple students can take multiple compulsory courses and vice versa in this case.



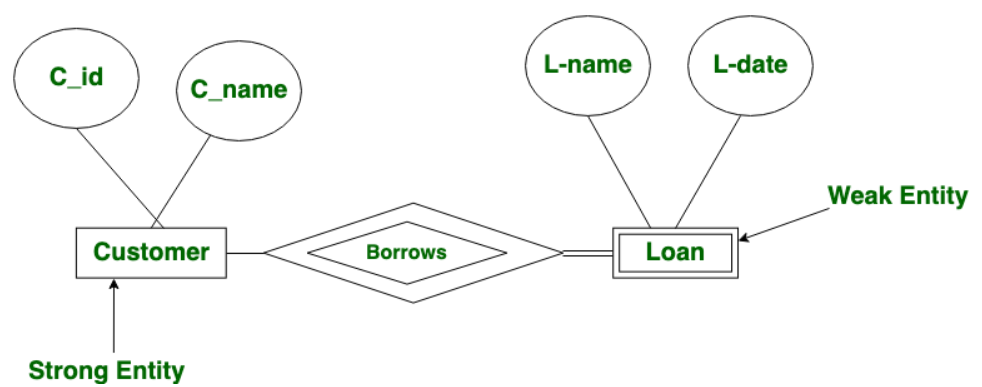
* + Here a employee may have multiple dependents or no one as dependents is a weak entity that doesn’t have primary key and also the relationship here is weak



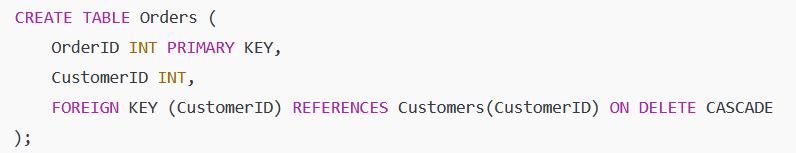
* Some general symbols that are used in the ER-Model



* Strong entity is one which has a primary key while the weak entity doesn’t have it.
* The weak entity is represented by the double rectangle box and if the relationship is weak then it is represented like



* ON DELETE CASCADE is a referential action in SQL used when defining a foreign key constraint. It ensures that when a record in the parent table (the referenced table) is deleted, all corresponding records in the child table (the referencing table) are automatically deleted as well.



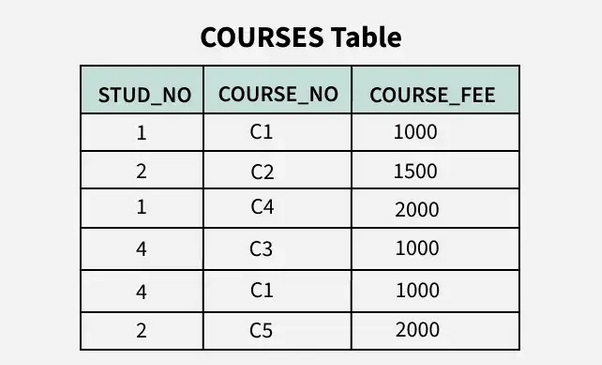
* Normalization is a process in database design that organizes data to reduce redundancy and improve data integrity. There are several **Normal Forms (NF)**, and each one builds on the previous one.
  + 1NF -table is in **1NF** if it contains only **atomic values**, meaning each field contains only one value. It should also have a unique row identifier (primary key) for each record.

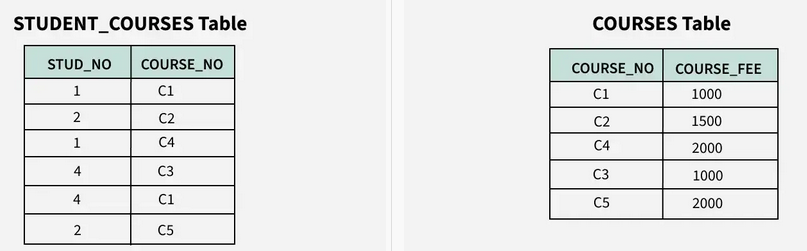
| **CustomerID** | **CustomerName** | **Products** |
| --- | --- | --- |
| 1 | John | Laptop, Phone |
| 2 | Jane | Tablet, Headphones |

Making the values atomic 🡪

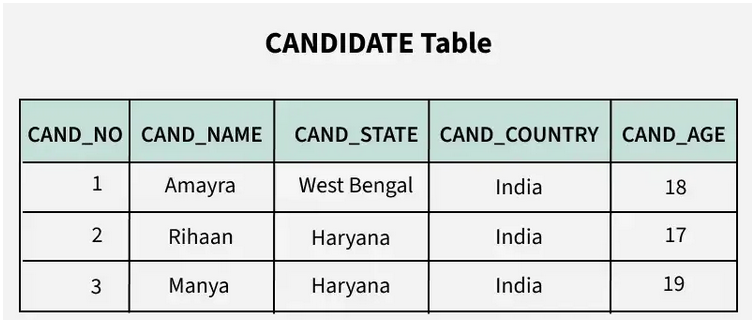
| **CustomerID** | **CustomerName** | **Product** |
| --- | --- | --- |
| 1 | John | Laptop |
| 1 | John | Phone |
| 2 | Jane | Tablet |
| 2 | Jane | Headphones |

* + A table is in **2NF** if it is in **1NF** and all non-key attributes (columns) are **fully dependent** on the primary key, i.e., there is no partial dependency (where a non-key column depends on part of the primary key).





* + A table is in **3NF** if it is in **2NF** and there are no **transitive dependencies**. This means non-key columns should not depend on other non-key columns.





* **Stored procedure** is a set of SQL statements that are stored and executed in the database. Stored procedures allow you to encapsulate logic, making your database operations more efficient and reusable.
  + Basic syntax:

DELIMITER $$ -> (it can be && or some other operator)

CREATE PROCEDURE procedure\_name (parameters)

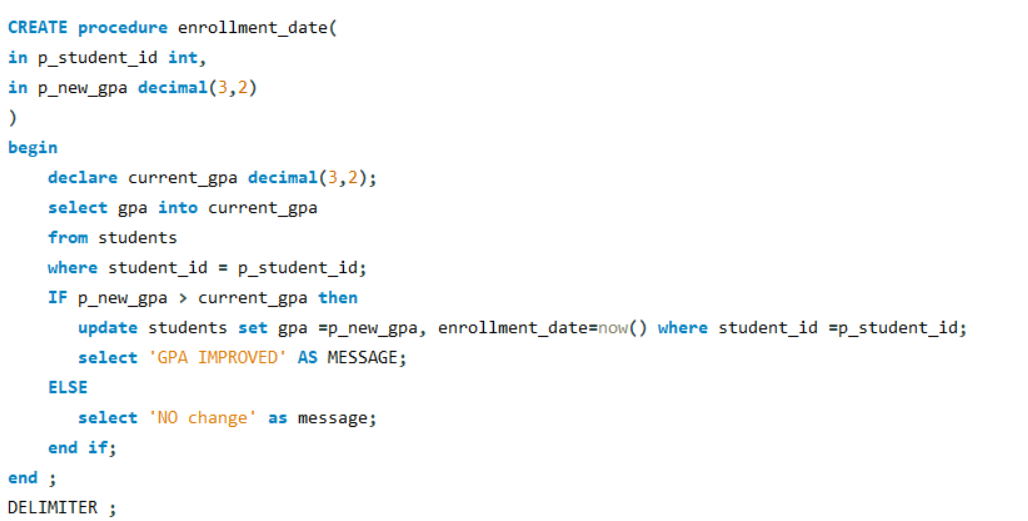
BEGIN

-- SQL statements go here

END$$

DELIMITER;

* Here we created a procedure



* We have created a procedure enrolment\_date that takes p\_student\_id with p\_new\_gpa as input when called and after it starts as a transaction and it declares another variable current\_gpa and creating a variable current\_gpa from gpa column from students table where student\_id=p\_student\_id and if p\_new\_gpa > current\_gpa then we will update students and set gpa= new one and update enrolment date with date and timestamp where student\_id is same as current input. And then show message GPA IMPROVED if not then just print message NO CHANGE .
* DECLARE is used to declare the variable.
* INTO is used to copy the variable.
* End if we are using at last 2nd line denotes that we have to close the if statement as we are using the if statement in an procedure.
* Everything we writes between the begin and end acts like a single transaction going on.
* DELIMITER is used so that multiple line are executed in a single line as a single transaction.
* SELECT ‘GPA IMPROVED ’ AS MESSAGE will simply print GPA IMPROVED in terminal if the specific condition is met.
* If we want to use some conditions like If else or some programmatic logic we use procedures
* IN SQL there is nothing by default just like function we have to use procedure to create one.
* CALL procedure\_name (params);
  + this is used to call a procedure.
* NOW(): Returns the current date and time in the format YYYY-MM-DD HH:MM:SS.
  + Syntax: SELECT NOW();
  + Use: When you need both the date and time (e.g., logging timestamps).
* CURDATE(): Returns only the current date in the format YYYY-MM-DD.
  + Syntax: SELECT CURDATE();
  + Use: When you only need the date (e.g., storing a creation date).
* ACID PROPERTIES:
  + ACID refers to the Atomicity,Consistency,Isolation,Durability
  + ATOMICITY: it says that either all the transactions are executed or none there is no partial condition.
    - E.g.either 100 records are inserted or no record .there can’t be a condition where 50 or 60 records are inserted.
  + CONSISTENCY: it says that database must be consistent there should be no intermediary phase. Means database should be consistent before and after the transaction
    - E.g. In the bank scenario, if a rule exists that no account balance can go below zero, Consistency ensures that no transaction violates this rule. If a transaction tries to withdraw more money than available in an account, it will not proceed and will maintain the integrity of the database.
  + ISLOATION: it ensures that concurrent transactions do not interfere with each other. Each transaction should execute independently, even if other transactions are running at the same time. It ensures that the result of a transaction is not visible to others until it is complete.
    - E.g. Imagine two people withdrawing money from the same bank account at the same time. Isolation ensures that one transaction is fully completed before the other one starts, preventing issues like one withdrawal being processed before the other.
  + DURABILITY: Durability ensures that once a transaction has been committed, it will persist in the database, even if there is a system crash. The data will not be lost and will be recoverable.
    - E.g. If you transfer money from Account A to Account B and the system crashes right after the transaction is complete, Durability ensures that the transfer will be saved and the updated balances will still exist once the system recovers.
  + ROLLBACK is always written into the procedure and if any error comes between the begin and end of the transaction everything is erased and back to normal.