Part – A (SQL)

Schema

Database Name: Branch_DIV_Rollno (Example: CSE_4A_101 or Bsc_Hons_101)
Note: Create all the tables under above database with design mode only.

Table 1: STUDENT

Column Name	Data Type	Constraints
StudentID	INT	Primary Key
StuName	VARCHAR(100)	NOT NULL
StuEmail	VARCHAR(100)	NULL
StuPhone	VARCHAR(15)	NULL
StuDepartment	VARCHAR(50)	NOT NULL
StuDateOfBirth	DATE	NOT NULL
StuEnrollmentYear	INT	NOT NULL

Student ID	StuName	StuEmail	StuPhone	Stu Department	Stu DOB	Stu Enrollment Year
1	Raj Patel	raj @univ.edu	9876543210	CSE	2003-05-15	2021
2	Priya Shah	priya@univ.edu	9876543211	IT	2002-08-22	2020
3	Amit Kumar	amit@univ.edu	9876543212	CSE	2003-11-10	2021
4	Sneha Desai	sneha@univ.edu	9876543213	ECE	2004-02-18	2022
5	Rohan Mehta	rohan@univ.edu	9876543214	IT	2003-07-25	2021
6	Kavita Joshi	kavita@univ.edu	9876543215	CSE	2002-12-30	2020
7	Arjun Verma	arjun@univ.edu	9876543216	MECH	2003-04-08	2021
8	Pooja Rao	pooja@univ.edu	9876543217	ECE	2004-06-12	2022

Table 2: COURSE

Column Name	Data Type	Constraints
CourseID	VARCHAR(10)	Primary Key
CourseName	VARCHAR(100)	NOT NULL
CourseCredits	INT	NOT NULL
CourseDepartment	VARCHAR(50)	NOT NULL
CourseSemester	INT	NOT NULL

CourseID	CourseName	Course Credits	Course Department	Course Semester
CS101	Programming Fundamentals	4	CSE	1
CS201	Data Structures	4	CSE	3
CS301	Database Management Systems	4	CSE	5
IT101	Web Technologies	3	IT	1
IT201	Software Engineering	4	IT	3
EC101	Digital Electronics	3	ECE	1
EC201	Microprocessors	4	ECE	3
ME101	Engineering Mechanics	4	MECH	1

CS202	Operating Systems	4	CSE	4
CS302	Artificial Intelligence	3	CSE	6

Table 3: FACULTY

Column Name	Data Type	Constraints
FacultyID	INT	Primary Key
FacultyName	VARCHAR(100)	NOT NULL
FacultyEmail	VARCHAR(100)	NULL
FacultyDepartment	VARCHAR(50)	NOT NULL
FacultyDesignation	VARCHAR(50)	NOT NULL
FacultyJoiningDate	DATE	NOT NULL

FacultyID	Faculty Name	FacultyEmail	Faculty Department	Faculty Designation	Faculty JoiningDate
101	Dr. Sheth	Sheth@univ.edu	CSE	Professor	2010-07-15
102	Prof. Gupta	gupta@univ.edu	IT	Associate Prof	2012-08-20
103	Dr. Patel	patel@univ.edu	CSE	Assistant Prof	2015-06-10
104	Dr. Singh	singh@univ.edu	ECE	Professor	2008-03-25
105	Prof. Reddy	reddy@univ.edu	IT	Assistant Prof	2018-01-15
106	Dr. Iyer	iyer@univ.edu	MECH	Associate Prof	2013-09-05
107	Prof. Nair	nair@univ.edu	CSE	Assistant Prof	2019-07-20

Table 4: ENROLLMENT

Column Name	Data Type	Constraints
EnrollmentID	INT	Primary Key, Auto Increment (1,1)
StudentID	INT	Foreign Key (STUDENT)
CourseID	VARCHAR(10)	Foreign Key (COURSE)
EnrollmentDate	DATE	NULL
Grade	VARCHAR(2)	NULL
EnrollmentStatus	VARCHAR(20)	NOT NULL, CHECK (EnrollmentStatus IN ('Active', 'Completed', 'Dropped'))

StudentID	CourseID	EnrollmentDate	Grade	EnrollmentStatus
1	CS101	2021-07-01	A	Completed
1	CS201	2022-01-05	B+	Completed
1	CS301	2023-07-01	NULL	Active
2	IT101	2020-07-01	A	Completed
2	IT201	2021-07-01	A-	Completed
3	CS101	2021-07-01	B	Completed
3	CS201	2022-01-05	A	Completed
4	EC101	2022-07-01	B+	Completed
5	IT101	2021-07-01	A	Completed
6	CS201	2021-01-05	A	Completed
1	CS302	2023-07-01	NULL	Active
2	IT201	2022-01-05	NULL	Dropped

Table 5: COURSE_ASSIGNMENT

Column Name	Data Type	Constraints
AssignmentID	INT	Primary Key, Auto Increment
CourseID	VARCHAR(10)	Foreign Key (COURSE)
FacultyID	INT	Foreign Key (FACULTY)
Semester	INT	NOT NULL
Year	INT	NOT NULL
ClassRoom	VARCHAR(20)	NOT NULL

CourseID	FacultyID	Semester	Year	ClassRoom
CS101	103	1	2024	A-301
CS201	101	3	2024	B-205
CS301	101	5	2024	A-401
IT101	102	1	2024	C-102
IT201	105	3	2024	C-205
EC101	104	1	2024	D-101
EC201	104	3	2024	D-203
ME101	106	1	2024	E-101
CS202	107	4	2024	A-305
CS302	101	6	2024	B-401

Lab-1 SQL Concepts Revision

Part – A

1. Retrieve all unique departments from the STUDENT table.
2. Insert a new student record into the STUDENT table.
   (9, 'Neha Singh', 'neha.singh@univ.edu', '9876543218', 'IT', '2003-09-20', 2021)
3. Change the Email of student 'Raj Patel' to 'raj.p@univ.edu'. (STUDENT table)
4. Add a new column 'CGPA' with datatype DECIMAL(3,2) to the STUDENT table.
5. Retrieve all courses whose CourseName starts with 'Data'. (COURSE table)
6. Retrieve all students whose Name contains 'Shah'. (STUDENT table)
7. Display all Faculty Names in UPPERCASE. (FACULTY table)
8. Find all faculty who joined after 2015. (FACULTY table)
9. Find the SQUARE ROOT of Credits for the course 'Database Management Systems'. (COURSE table)
10. Find the Current Date using SQL Server in-built function.
11. Find the top 3 students who enrolled earliest (by EnrollmentYear). (STUDENT table)
12. Find all enrollments that were made in the year 2022. (ENROLLMENT table)
13. Find the number of courses offered by each department. (COURSE table)
14. Retrieve the CourseID which has more than 2 enrollments. (ENROLLMENT table)
15. Retrieve all the student name with their enrollment status. (STUDENT & ENROLLMENT table)
16. Select all student names with their enrolled course names. (STUDENT, COURSE, ENROLLMENT table)
17. Create a view called 'ActiveEnrollments' showing only active enrollments with student name and course name. (STUDENT, COURSE, ENROLLMENT, table)
18. Retrieve the student's name who is not enrol in any course using subquery. (STUDENT, ENROLLMENT TABLE)
19. Display course name having second highest credit. (COURSE table)

Part – B

20. Retrieve all courses along with the total number of students enrolled. (COURSE, ENROLLMENT table)
21. Retrieve the total number of enrollments for each status, showing only statuses that have more than 2 enrollments. (ENROLLMENT table)
22. Retrieve all courses taught by 'Dr. Sheth' and order them by Credits. (FACULTY, COURSE, COURSE_ASSIGNMENT table)

Part – C

23. List all students who are enrolled in more than 3 courses. (STUDENT, ENROLLMENT table)
24. Find students who have enrolled in both 'CS101' and 'CS201' Using Sub Query. (STUDENT, ENROLLMENT table)
25. Retrieve department-wise count of faculty members along with their average years of experience (calculate experience from JoiningDate). (Faculty table)

Lab-2 Stored Procedure

Part – A

1. **INSERT Procedures:** Create stored procedures to insert records into STUDENT tables (SP_INSERT_STUDENT)

StuID	Name	Email	Phone	Department	DOB	EnrollmentYear
10	Harsh Parmar	harsh@univ.edu	9876543218	CSE	2005-09-18	2023
20	Om Patel	om@univ.edu	9876543211	IT	2002-08-22	2022

2. **INSERT Procedures:** Create stored procedures to insert records into COURSE tables (SP_INSERT_COURSE)

CourseID	CourseName	Credits	Dept	Semester
CS330	Computer Networks	4	CSE	5
EC120	Electronic Circuits	3	ECE	2

3. **UPDATE Procedures:** Create stored procedure SP_UPDATE_STUDENT to update Email and Phone in STUDENT table. (Update using studentID)
4. **DELETE Procedures:** Create stored procedure SP_DELETE_STUDENT to delete records from STUDENT where Student Name is Om Patel.
5. **SELECT BY PRIMARY KEY:** Create stored procedures to select records by primary key (SP_SELECT_STUDENT_BY_ID) from Student table.
6. Create a stored procedure that shows details of the first 5 students ordered by EnrollmentYear.

Part – B

7. Create a stored procedure which displays faculty designation-wise count.
8. Create a stored procedure that takes department name as input and returns all students in that department.

Part – C

9. Create a stored procedure which displays department-wise maximum, minimum, and average credits of courses.
10. Create a stored procedure that accepts StudentID as parameter and returns all courses the student is enrolled in with their grades.

Lab-3	**Advanced Stored Procedure**
	Part – A 1. Create a stored procedure that accepts a date and returns all faculty members who joined on that date. 2. Create a stored procedure for ENROLLMENT table where user enters either StudentID and returns EnrollmentID, EnrollmentDate, Grade, and Status. 3. Create a stored procedure that accepts two integers (min and max credits) and returns all courses whose credits fall between these values. 4. Create a stored procedure that accepts Course Name and returns the list of students enrolled in that course. 5. Create a stored procedure that accepts Faculty Name and returns all course assignments. 6. Create a stored procedure that accepts Semester number and Year, and returns all course assignments with faculty and classroom details. **Part – B** 7. Create a stored procedure that accepts the first letter of Status ('A', 'C', 'D') and returns enrollment details. 8. Create a stored procedure that accepts either Student Name OR Department Name and returns student data accordingly. 9. Create a stored procedure that accepts CourseID and returns all students enrolled grouped by enrollment status with counts. **Part – C** 10. Create a stored procedure that accepts a year as input and returns all courses assigned to faculty in that year with classroom details. 11. Create a stored procedure that accepts From Date and To Date and returns all enrollments within that range with student and course details. 12. Create a stored procedure that accepts FacultyID and calculates their total teaching load (sum of credits of all courses assigned).

Lab-4	**UDF**
	Part – A 1. Write a scalar function to print "Welcome to DBMS Lab". 2. Write a scalar function to calculate simple interest. 3. Function to Get Difference in Days Between Two Given Dates 4. Write a scalar function which returns the sum of Credits for two given CourseIDs. 5. Write a function to check whether the given number is ODD or EVEN. 6. Write a function to print number from 1 to N. (Using while loop) 7. Write a scalar function to calculate factorial of total credits for a given CourseID. 8. Write a scalar function to check whether a given EnrollmentYear is in the past, current or future (Case statement) 9. Write a table-valued function that returns details of students whose names start with a given letter. 10. Write a table-valued function that returns unique department names from the STUDENT table. **Part – B** 11. Write a scalar function that calculates age in years given a DateOfBirth. 12. Write a scalar function to check whether given number is palindrome or not. 13. Write a scalar function to calculate the sum of Credits for all courses in the 'CSE' department. 14. Write a table-valued function that returns all courses taught by faculty with a specific designation. **Part – C** 15. Write a scalar function that accepts StudentID and returns their total enrolled credits (sum of credits from all active enrollments).

	16. Write a scalar function that accepts two dates (joining date range) and returns the count of faculty who joined in that period.

Lab-5	**Cursor**
	Part – A 1. Create a cursor Course_Cursor to fetch all rows from COURSE table and display them. 2. Create a cursor Student_Cursor_Fetch to fetch records in form of StudentID_StudentName (Example: 1_Raj Patel). 3. Create a cursor to find and display all courses with Credits greater than 3. 4. Create a cursor to display all students who enrolled in year 2021 or later. 5. Create a cursor Course_CursorUpdate that retrieves all courses and increases Credits by 1 for courses with Credits less than 4. 6. Create a Cursor to fetch Student Name with Course Name (Example: Raj Patel is enrolled in Database Management System) 7. Create a cursor to insert data into new table if student belong to 'CSE' department. (create new table CSEStudent with relevant columns) **Part – B** 8. Create a cursor to update all NULL grades to 'F' for enrollments with Status 'Completed' 9. Cursor to show Faculty with Course they teach (EX: Dr. Sheth teaches Data structure) **Part – C** 10. Cursor to calculate total credits per student (Example: Raj Patel has total credits = 15)

Lab-6	**Trigger (After trigger)**
	Table : Log(LogMessage varchar(100), logDate Datetime) **Part – A** 1. Create trigger for printing appropriate message after student registration. 2. Create trigger for printing appropriate message after faculty deletion. 3. Create trigger for monitoring all events on course table. (print only appropriate message) 4. Create trigger for logging data on new student registration in Log table. 5. Create trigger for auto-uppercasing faculty names. 6. Create trigger for calculating faculty experience (Note: Add required column in faculty table) **Part – B** 7. Create trigger for auto-stamping enrollment dates. 8. Create trigger for logging data After course assignment - log course and faculty detail. **Part - C** 9. Create trigger for updating student phone and print the old and new phone number. 10. Create trigger for updating course credit log old and new credits in log table.

Lab-7	**Trigger (Instead of trigger)**
	Table : Log(LogMessage varchar(100), logDate Datetime) **Part – A** 1. Create trigger for blocking student deletion. 2. Create trigger for making course read-only. 3. Create trigger for preventing faculty removal.

	4. Create instead of trigger to log all operations on COURSE (INSERT/UPDATE/DELETE) into Log table. (Example: INSERT/UPDATE/DELETE operations are blocked for you in course table) 5. Create trigger to Block student to update their enrollment year and print message 'students are not allowed to update their enrollment year' 6. Create trigger for student age validation (Min 18). **Part – B** 7. Create trigger for unique faculty's email check. 8. Create trigger for preventing duplicate enrollment. **Part - C** 9. Create trigger to Allow enrolment in month from Jan to August, otherwise print message enrolment closed. 10. Create trigger to Allow only grade change in enrollment (block other updates)

Lab-8	**Exception Handling**
	Part – A 1. Handle Divide by Zero Error and Print message like: Error occurs that is - Divide by zero error. 2. Try to convert string to integer and handle the error using try…catch block. 3. Create a procedure that prints the sum of two numbers: take both numbers as integer & handle exception with all error functions if any one enters string value in numbers otherwise print result. 4. Handle a Primary Key Violation while inserting data into student table and print the error details such as the error message, error number, severity, and state. 5. Throw custom exception using stored procedure which accepts StudentID as input & that throws Error like no StudentID is available in database. 6. Handle a Foreign Key Violation while inserting data into Enrollment table and print appropriate error message. **Part – B** 7. Handle Invalid Date Format 8. Procedure to Update faculty's Email with Error Handling. 9. Throw custom exception that throws error if the data is invalid. **Part – C** 10. Write a script that checks if a faculty's salary is NULL. If it is, use RAISERROR to show a message with a severity of 16. (Note: Do not use any table)