Part – A (SQL)

Lab-1 **SQL Concepts Revision**

Database Name: Branch_DIV_Rollno (Example: CSE_3A_101 or Bsc_Hons_101)

Note: Create all the tables under above database with design mode only.

Table1: Artists (Artist_id (PK), Artist_name)

Artist_id	Artist_name
1	Aparshakti Khurana
2	Ed Sheeran
3	Shreya Ghoshal
4	Arijit Singh
5	Tanishk Bagchi

Table2: Albums (Album_id(PK), Album_title, Artist_id(FK), Release_year)

Album_id	Album_title	Artist_id	Release_year
1001	Album1	1	2019
1002	Album2	2	2015
1003	Album3	3	2018
1004	Album4	4	2020
1005	Album5	2	2020
1006	Album6	1	2009

Table3: Songs (Song_id, Song_title, Duration (in minutes), Genre, Album_id(FK))

Song_id	Song_title	Duration	Genre	Album_id
101	Zaroor	2.55	Feel good	1001
102	Espresso	4.10	Rhythmic	1002
103	Shayad	3.20	Sad	1003
104	Roar	4.05	Pop	1002
105	Everybody Talks	3.35	Rhythmic	1003
106	Dwapara	3.54	Dance	1002
107	Sa Re Ga Ma	4.20	Rhythmic	1004
108	Tauba	4.05	Rhythmic	1005
109	Perfect	4.23	Pop	1002
110	Good Luck	3.55	Rhythmic	1004

Part – A

1. Retrieve a unique genre of songs.
2. Find top 2 albums released before 2010.
3. Insert Data into the Songs Table. (1245, ‘Zaroor’, 2.55, ‘Feel good’, 1005)
4. Change the Genre of the song ‘Zaroor’ to ‘Happy’
5. Delete an Artist ‘Ed Sheeran’
6. Add a New Column for Rating in Songs Table. [Ratings decimal(3,2)]
7. Retrieve songs whose title starts with 'S'.
8. Retrieve all songs whose title contains 'Everybody'.
9. Display Artist Name in Uppercase.
10. Find the Square Root of the Duration of a Song ‘Good Luck’
11. Find Current Date.
12. Find the number of albums for each artist.
13. Retrieve the Album_id which has more than 5 songs in it.
14. Retrieve all songs from the album 'Album1'. (using Subquery)

15. Retrieve all albums name from the artist 'Aparshakti Khurana' (using Subquery)
16. Retrieve all the song titles with its album title.
17. Find all the songs which are released in 2020.
18. Create a view called 'Fav_Songs' from the songs table having songs with song_id 101-105.
19. Update a song name to 'Jannat' of song having song_id 101 in Fav_Songs view.
20. Find all artists who have released an album in 2020.
21. Retrieve all songs by Shreya Ghoshal and order them by duration.

Part – B

22. Retrieve all song titles by artists who have more than one album.
23. Retrieve all albums along with the total number of songs.
24. Retrieve all songs and release year and sort them by release year.
25. Retrieve the total number of songs for each genre, showing genres that have more than 2 songs.
26. List all artists who have albums that contain more than 3 songs.

Part – C

27. Retrieve albums that have been released in the same year as 'Album4'
28. Find the longest song in each genre
29. Retrieve the titles of songs released in albums that contain the word 'Album' in the title.
30. Retrieve the total duration of songs by each artist where total duration exceeds 15 minutes.

Lab-2 Stored Procedure

Person

Column_Name	DataType	Constraints
PersonID	Int	Primary Key, Auto Increment
FirstName	Varchar (100)	Not Null
LastName	Varchar (100)	Not Null
Salary	Decimal (8,2)	Not Null
JoiningDate	Datetime	Not Null
DepartmentID	Int	Foreign Key, Null
DesignationID	Int	Foreign Key, Null

Department

Column_Name	DataType	Constraints
DepartmentID	Int	Primary Key
DepartmentName	Varchar (100)	Not Null, Unique

Designation

Column_Name	DataType	Constraints
DesignationID	Int	Primary Key
DesignationName	Varchar (100)	Not Null, Unique

PersonID	FirstName	LastName	Salary	JoiningDate	DepartmentID	DesignationID
101	Rahul	Anshu	56000	01-01-1990	1	12
102	Hardik	Hinsu	18000	25-09-1990	2	11
103	Bhavin	Kamani	25000	14-05-1991	*NULL*	11
104	Bhoomi	Patel	39000	20-02-2014	1	13
105	Rohit	Rajgor	17000	23-07-1990	2	15
106	Priya	Mehta	25000	18-10-1990	2	*NULL*
107	Neha	Trivedi	18000	20-02-2014	3	15

DepartmentID	DepartmentName
1	Admin
2	IT
3	HR
4	Account

DesignationID	DesignationName
11	Jobber
12	Welder
13	Clerk
14	Manager
15	CEO

From the above given tables create Stored Procedures:

Part – A

1. Department, Designation & Person Table's INSERT, UPDATE & DELETE Procedures.
2. Department, Designation & Person Table's SELECTBYPRIMARYKEY
3. Department, Designation & Person Table's (If foreign key is available then do write join and take columns on select list)
4. Create a Procedure that shows details of the first 3 persons.

Part – B

5. Create a Procedure that takes the department name as input and returns a table with all workers working in that department.
6. Create Procedure that takes department name & designation name as input and returns a table with worker's first name, salary, joining date & department name.
7. Create a Procedure that takes the first name as an input parameter and display all the details of the worker with their department & designation name.
8. Create Procedure which displays department wise maximum, minimum & total salaries.
9. Create Procedure which displays designation wise average & total salaries.

Part – C

10. Create Procedure that Accepts Department Name and Returns Person Count.
11. Create a procedure that takes a salary value as input and returns all workers with a salary greater than input salary value along with their department and designation details.
12. Create a procedure to find the department(s) with the highest total salary among all departments.
13. Create a procedure that takes a designation name as input and returns a list of all workers under that designation who joined within the last 10 years, along with their department.
14. Create a procedure to list the number of workers in each department who do not have a designation assigned.
15. Create a procedure to retrieve the details of workers in departments where the average salary is above 12000.

Lab-3 Advanced Stored Procedure

Departments			
ColumnName	**DataType**	**AN | NN**	**Remarks**
DepartmentID	Int	NN	Primary Key
DepartmentName	Varchar(100)	NN	
ManagerID	Int	NN	
Location	Varchar(100)	NN	

Employee			
Column Name	**Data Type**	**AN | NN**	**Remarks**
EmployeeID	Int	NN	Primary Key
FirstName	Varchar(100)	NN	
LastName	Varchar(100)	NN	
DoB	Datetime	NN	
Gender	Varchar(50)	NN	
HireDate	Datetime	NN	
DeptID	Int	NN	Foreign Key
Salary	Decimal(10,2)	NN	

Projects			
ColumnName	**DataType**	**AN | NN**	**Remarks**
ProjectID	Int	NN	Primary Key
ProjectName	Varchar(100)	NN	
StartDate	Datetime	NN	
EndDate	Datetime	NN	
DepartmentID	Int	NN	Foreign Key

DepartmentID	DepartmentName	ManagerID	Location
1	IT	101	New York
2	HR	102	San Francisco
3	Finance	103	Los Angeles
4	Admin	104	Chicago
5	Marketing	105	Miami

EmployeeID	FirstName	LastName	DoB	Gender	HireDate	DeptID	Salary
101	John	Doe	1985-04-12	Male	2010-06-15	1	75000.00
102	Jane	Smith	1990-08-24	Female	2015-03-10	2	60000.00
103	Robert	Brown	1982-12-05	Male	2008-09-25	3	82000.00
104	Emily	Davis	1988-11-11	Female	2012-07-18	4	58000.00
105	Michael	Wilson	1992-02-02	Male	2018-11-30	5	67000.00

ProjectID	ProjectName	StartDate	EndDate	DepartmentID
201	Project Alpha	2022-01-01	2022-12-31	1
202	Project Beta	2023-03-15	2024-03-14	2
203	Project Gamma	2021-06-01	2022-05-31	3
204	Project Delta	2020-10-10	2021-10-09	4
205	Project Epsilon	2024-04-01	2025-03-31	5

Part – A

1. Create Stored Procedure for Employee table As User enters either First Name or Last Name and based on this you must give EmployeeID, DOB, Gender & Hiredate.
2. Create a Procedure that will accept Department Name and based on that gives employees list who belongs to that department.

	3. Create a Procedure that accepts Project Name & Department Name and based on that you must give all the project related details. 4. Create a procedure that will accepts any integer and if salary is between provided integer, then those employee list comes in output. 5. Create a Procedure that will accepts a date and gives all the employees who all are hired on that date. **Part – B** 6. Create a Procedure that accepts Gender's first letter only and based on that employee details will be served. 7. Create a Procedure that accepts First Name or Department Name as input and based on that employee data will come. 8. Create a procedure that will accepts location, if user enters a location any characters, then he/she will get all the departments with all data. **Part – C** 9. Create a procedure that will accepts From Date & To Date and based on that he/she will retrieve Project related data. 10. Create a procedure in which user will enter project name & location and based on that you must provide all data with Department Name, Manager Name with Project Name & Starting Ending Dates.

Lab-4	**UDF**
	Note: for Table valued function use tables of Lab-2 **Part – A** 1. Write a function to print "hello world". 2. Write a function which returns addition of two numbers. 3. Write a function to check whether the given number is ODD or EVEN. 4. Write a function which returns a table with details of a person whose first name starts with B. 5. Write a function which returns a table with unique first names from the person table. 6. Write a function to print number from 1 to N. (Using while loop) 7. Write a function to find the factorial of a given integer. **Part – B** 8. Write a function to compare two integers and return the comparison result. (Using Case statement) 9. Write a function to print the sum of even numbers between 1 to 20. 10. Write a function that checks if a given string is a palindrome **Part – C** 11. Write a function to check whether a given number is prime or not. 12. Write a function which accepts two parameters start date & end date, and returns a difference in days. 13. Write a function which accepts two parameters year & month in integer and returns total days each year. 14. Write a function which accepts departmentID as a parameter & returns a detail of the persons. 15. Write a function that returns a table with details of all persons who joined after 1-1-1991.

Lab-5	**Trigger**

PersonInfo

Column_Name	DataType	Constraints
PersonID	Int	Primary Key
PersonName	Varchar (100)	Not Null
Salary	Decimal (8,2)	Not Null
JoiningDate	Datetime	Allow Null

City	Varchar (100)	Not Null
Age	Int	Allow Null
BirthDate	Datetime	Not Null

PersonLog		
Column_Name	**DataType**	**Constraints**
PLogID	Int	Primary Key, Auto increment
PersonID	Int	Not Null
PersonName	Varchar (250)	Not Null
Operation	Varchar (50)	Not Null
UpdateDate	Datetime	Not Null

From the above given tables perform the following queries:

Part – A

1. Create a trigger that fires on INSERT, UPDATE and DELETE operation on the PersonInfo table to display a message "Record is Affected."
2. Create a trigger that fires on INSERT, UPDATE and DELETE operation on the PersonInfo table. For that, log all operations performed on the person table into PersonLog.
3. Create an INSTEAD OF trigger that fires on INSERT, UPDATE and DELETE operation on the PersonInfo table. For that, log all operations performed on the person table into PersonLog.
4. Create a trigger that fires on INSERT operation on the PersonInfo table to convert person name into uppercase whenever the record is inserted.
5. Create trigger that prevent duplicate entries of person name on PersonInfo table.
6. Create trigger that prevent Age below 18 years.

Part – B

7. Create a trigger that fires on INSERT operation on person table, which calculates the age and update that age in Person table.
8. Create a Trigger to Limit Salary Decrease by a 10%.

Part – C

9. Create Trigger to Automatically Update JoiningDate to Current Date on INSERT if JoiningDate is NULL during an INSERT.
10. Create DELETE trigger on PersonLog table, when we delete any record of PersonLog table it prints 'Record deleted successfully from PersonLog'.

Lab-6 **Cursor**

Products		
Column_Name	**DataType**	**Constraints**
Product_id	Int	Primary Key
Product_Name	Varchar (250)	Not Null
Price	Decimal (10,2)	Not Null

Products		
Product_id	**Product_Name**	**Price**
1	Smatphone	35000
2	Laptop	65000
3	Headphones	5500
4	Television	85000
5	Gaming Console	32000

From the above given tables perform the following queries:

Part - A

1. Create a cursor Product_Cursor to fetch all the rows from a products table.
2. Create a cursor Product_Cursor_Fetch to fetch the records in form of ProductID_ProductName. (Example: 1_Smartphone)
3. Create a Cursor to Find and Display Products Above Price 30,000.
4. Create a cursor Product_CursorDelete that deletes all the data from the Products table.

Part – B

5. Create a cursor Product_CursorUpdate that retrieves all the data from the products table and increases the price by 10%.
6. Create a Cursor to Rounds the price of each product to the nearest whole number.

Part – C

7. Create a cursor to insert details of Products into the NewProducts table if the product is "Laptop" (Note: Create NewProducts table first with same fields as Products table)
8. Create a Cursor to Archive High-Price Products in a New Table (ArchivedProducts), Moves products with a price above 50000 to an archive table, removing them from the original Products table.

Lab-7 Exception Handling

Customers

Column_Name	DataType	Constraints
Customer_id	Int	Primary Key
Customer _Name	Varchar (250)	Not Null
Email	Varchar (50)	Unique

Orders

Column_Name	DataType	Constraints
Order_id	Int	Primary Key
Customer_id	Int	Foreign Key
Order_date	date	Not Null

From the above given tables perform the following queries:

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 customers 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 Customer_id as input & that throws Error like no Customer_id is available in database.

Part – B

6. Handle a Foreign Key Violation while inserting data into Orders table and print appropriate error message.
7. Throw custom exception that throws error if the data is invalid.
8. Create a Procedure to Update Customer's Email with Error Handling

Part – C

9. Create a procedure which prints the error message that "The Customer_id is already taken. Try another one".

	10. Handle Duplicate Email Insertion in Customers Table.

	Part – B (MongoDB)
Lab-8	**createcollection, dropcollection and insert method**

Perform following queries using use, drop, createcollection, dropcollection, insertOne and insertMany method.

Part - A

1. Create a new database named “Darshan”.
2. Create another new database named “DIET”.
3. List all databases.
4. Check the current database.
5. Drop “DIET” database.
6. Create a collection named “Student” in the “Darshan” database.
7. Create a collection named “Department” in the “Darshan” database.
8. List all collections in the “Darshan” database.
9. Insert a single document using insertOne into “Department” collection. (Dname:’CE’, HOD:’Patel’)
10. Insert two document using insertMany into “Department” collection. (Dname:’IT’ and Dname:’ICT’)
11. Drop a collection named “Department” from the “Darshan” database.
12. Insert a single document using insertOne into “Student” collection.
    (Fields are Name, City, Branch, Semester, Age) Insert your own data.
13. Insert three documents using insertMany into “Student” collection.
    (Fields are Name, City, Branch, Semester, Age) Insert your three friend’s data.
14. Check whether “Student” collection exists or not.
15. Check the stats of “Student” collection.
16. Drop the “Student” collection.
17. Create a collection named “Deposit”.
18. Insert following data in to “Deposit” collection.

Deposit

ACTNO	CNAME	BNAME	AMOUNT	CITY
101	ANIL	VRCE	1000.00	RAJKOT
102	SUNIL	AJNI	5000.00	SURAT
103	MEHUL	KAROLBAGH	3500.00	BARODA
104	MADHURI	CHANDI	1200.00	AHMEDABAD
105	PRMOD	M.G. ROAD	3000.00	SURAT
106	SANDIP	ANDHERI	2000.00	RAJKOT
107	SHIVANI	VIRAR	1000.00	SURAT
108	KRANTI	NEHRU PLACE	5000.00	RAJKOT

19. Display all the documents of “Deposit” collection.
20. Drop the “Deposit” collection.

Part – B

1. Create a new database named “Computer”.
2. Create a collection named “Faculty” in the “Computer” database.
3. Insert a below document using insertOne into “Faculty” collection.

Faculty

FID	FNAME	BNAME	SALARY	JDATE

1	ANIL	CE	10000	1-3-95

4. Insert below documents using insertMany into "Faculty" collection.

Faculty				
FID	FNAME	BNAME	SALARY	JDATE
2	SUNIL	CE	50000	4-1-96
3	MEHUL	IT	35000	17-11-95
4	MADHURI	IT	12000	17-12-95
5	PRMOD	CE	30000	27-3-96
6	SANDIP	CE	20000	31-3-96
7	SHIVANI	CE	10000	5-9-95
8	KRANTI	IT	50000	2-7-95

5. Display all the documents of "Faculty" collection.
6. Drop the "Faculty" collection.
7. Drop the "Computer" database.

Part – C (Perform following operation using UI)

1. Create a new database named "Computer".
2. Create a collection named "Faculty" in the "Computer" database.
3. Insert a below documents into "Faculty" collection.

Faculty				
FID	FNAME	BNAME	SALARY	JDATE
1	ANIL	CE	10000	1-3-95
2	SUNIL	CE	50000	4-1-96
3	MEHUL	IT	35000	17-11-95
4	MADHURI	IT	12000	17-12-95
5	PRMOD	CE	30000	27-3-96
6	SANDIP	CE	20000	31-3-96
7	SHIVANI	CE	10000	5-9-95
8	KRANTI	IT	50000	2-7-95

4. Display all the documents of "Faculty" collection.
5. Drop the "Faculty" collection.
6. Drop the "Computer" database.

Lab-9	**Find, limit, skip and sort method**
Mongo DB	**Perform following queries using find, limit, skip and sort method.** Create and Select Database Named: "**BANK_INFO"**

Deposit (Collection name)				
ACTNO	CNAME	BNAME	AMOUNT	ADATE
101	ANIL	VRCE	1000	1-3-95
102	SUNIL	AJNI	5000	4-1-96
103	MEHUL	KAROLBAGH	3500	17-11-95
104	MADHURI	CHANDI	1200	17-12-95
105	PRMOD	M.G. ROAD	3000	27-3-96
106	SANDIP	ANDHERI	2000	31-3-96
107	SHIVANI	VIRAR	1000	5-9-95
108	KRANTI	NEHRU PLACE	5000	2-7-95

From the above given collection perform the following queries using find, limit, skip and sort method:

Part - A

1. Retrieve/Display every document of Deposit collection.
2. Display only one document of Deposit collection. (Use: findOne())
3. Insert following document into Deposit collection. (Use: insertOne())

109	KIRTI	VIRAR	3000	3-5-97

4. Insert following documents into Deposit collection. (Use: insertMany())

110	MITALI	ANDHERI	4500	4-9-95
111	RAJIV	NEHRU PLACE	7000	2-10-98

5. Display all the documents of 'VIRAR' branch from Deposit collection.
6. Display all the documents of Deposit collection whose amount is between 3000 and 5000.
7. Display all the documents of Deposit collection whose amount is greater than 2000 and branch is VIRAR.
8. Display all the documents with CNAME, BNAME and AMOUNT fields from Deposit collection.
9. Display all the documents of Deposit collection on ascending order by CNAME.
10. Display all the documents of Deposit collection on descending order by BNAME.
11. Display all the documents of Deposit collection on ascending order by ACTNO and descending order by AMOUNT.
12. Display only two documents of Deposit collection.
13. Display 3rd document of Deposit collection.
14. Display 6th and 7th documents of Deposit collection.
15. Display the count of documents in Deposit collection.

Part- B

1. Insert following documents into "Student" collection. (Use: insertMany())

```
{ "_id": 1, "name": "John", "age": 30, "city": "New York", "isActive": true }
{ "_id": 2, "name": "Jane", "age": 25, "city": "Los Angeles", "isActive": false }
{ "_id": 3, "name": "Tom", "age": 35, "city": "Chicago", "isActive": true }
{ "_id": 4, "name": "Lucy", "age": 28, "city": "San Francisco", "isActive": true }
{ "_id": 5, "name": "David", "age": 40, "city": "Miami", "isActive": false }
{ "_id": 6, "name": "Eva", "age": 23, "city": "Boston", "isActive": true }
{ "_id": 7, "name": "Nick", "age": 38, "city": "Seattle", "isActive": false }
{ "_id": 8, "name": "Sophia", "age": 27, "city": "New York", "isActive": true }
{ "_id": 9, "name": "Liam", "age": 32, "city": "Los Angeles", "isActive": false }
{ "_id": 10, "name": "Olivia", "age": 29, "city": "San Diego", "isActive": true }
```

2. Display all documents of "Student" collection.
3. Display all documents of "Student" collection whose age is 30.
4. Display all documents of "Student" collection whose age is greater than 25.
5. Display all documents of "Student" collection whose name is "John" and age is 30.
6. Display all documents of "Student" collection whose age is not equal to 25.
7. Display all documents of "Student" collection whose age is equal to 25 or 30 or 35. (using $or as well as using $in).
8. Display all documents of "Student" collection whose name is "John" or age is 30.
9. Display all documents of "Student" collection whose name is "John" and city is New York.
10. Display name and age of students from "Student" collection whose name is "John" and city is New York.

Part – C

1. Display name of students from "Student" collection whose age is between to 25 and 35 and sort output by age in ascending order.

	2. Display all documents of "Student" collection and sort all the documents by name in ascending order and then by age in descending. 3. Display first five documents of "Student" collection. 4. Display fourth and fifth documents of "Student" collection. 5. Display the name of oldest student from "Student" collection. 6. Display all documents of "Student" collection in such a way that skip the first 2 documents and return the rest documents.

Lab-10	**update, delete and rename method**
Mongo DB	**Perform the following queries using update, delete, rename and createcollection method:** **Part – A (Use collection "Student" created in Lab-9)** 1. Update the age of John's to 31. 2. Update the city of all students from 'New York' to 'New Jersey'. 3. Set isActive to false for every student older than 35. 4. Increment the age of all students by 1 year. 5. Set the city of 'Eva' to 'Cambridge'. 6. Update 'Sophia's isActive status to false. 7. Update the city field of student aged below 30 to 'San Diego'. 8. Rename the age field to years for all documents. 9. Update 'Nick' to make him active (isActive = true). 10. Update all documents to add a new field country with the value 'USA'. 11. Update 'David's city to 'Orlando'. 12. Multiply the age of all students by 2. 13. Unset (remove) the city field for 'Tom'. 14. Add a new field premiumUser and to true for users older than 30. 15. Set isActive to true for 'Jane'. 16. Update isActive field of 'Lucy' to false. 17. Delete a document of 'Nick' from the collection. 18. Delete all students who are inactive (isActive = false). 19. Delete all students who live in 'New York'. 20. Delete all the students aged above 35. 21. Delete a student named ''Olivia'' from the collection. 22. Delete all the students whose age is below 25. 23. Delete the first student whose isActive field is true. 24. Delete all students from 'Los Angeles'. 25. Delete all students who have city field missing. 26. Rename 'city' field to 'location' for all documents. 27. Rename the name field to FullName for 'John'. 28. Rename the isActive field to status for all documents. 29. Rename age to yearsOld for everyone from 'San Francisco' student only. 30. Create a Capped Collection named "Employee" as per follows: a. Ecode and Ename are compulsory fields b. Datatype of EID is int, Ename is string, Age is int and City is string Insert following documents into above "Employee" collection.

{"Ecode": 1, "Ename": "John"}
{"Ecode ": 2, "Ename": "Jane", "age": 25, "city": "Los Angeles"}
{"Ecode ": 3, "Ename": "Tom", "age": 35}
{"Ecode ": 4, "Ename": "Lucy", "age": 28, "city": "San Francisco", "isActive": true}
{"Ename": "Dino"}

Part – B Create collection named "Student_data" and insert following 10 documents into it.

Student_data						
ROLLNO	**SNAME**	**DEPARTMENT**	**FEES**	**SEM**	**GENDER**	**CITY**
101	Vina	CE	15000	3	Female	Rajkot
102	Krisha	EC	8000	5	Female	Ahmedabad
103	Priti	Civil	12000	7	Female	Baroda
104	Mitul	CE	15000	3	Male	Rajkot
105	Keshav	CE	15000	3	Male	Jamnagar
106	Zarna	Civil	12000	5	Female	Ahmedabad
107	Nima	EE	9000	5	Female	Rajkot
108	Dhruv	Mechanical	10000	5	Male	Rajkot
109	Krish	Mechanical	10000	7	Male	Baroda
110	Zeel	EE	9000	3	Female	Jamnagar

From the above given "Student_data" collection perform the following queries:

1. Display Female students and belong to Rajkot city.
2. Display students not studying in 3rd sem.
3. Display students whose city is Jamnagar or Baroda. (use: IN)
4. Display first 2 students names who lives in Baroda.
5. Display Male students who studying in 3rd sem.
6. Display sname and city and fees of those students whose roll no is less than 105.
7. Update City of all students from 'Jamnagar' City and Department as 'CE' to 'Surat'.
8. Increase Fees by 500 where the Gender is not 'Female'. (Use: Not)
9. Set the Department of all students from 'EE' and in Sem 3 to 'Electrical'.
10. Update the Fees of students in 'Rajkot' who are male.
11. Change City to 'Vadodara' for students in Sem 5 and with fees less than 10000.
12. Delete all students where the City is 'Ahmedabad' or GENDER is 'Male'.
13. Delete students whose Rollno is not in the list [101, 105, 110].
14. Delete students from the 'Civil' department who are in Sem 5 or Sem 7.
15. Delete all students who are not in the cities 'Rajkot', 'Baroda', or 'Jamnagar'.
16. Delete students whose Rollno is between 105 and 108.
17. Rename the City field to LOCATION for all students.
18. Rename the Department field to Branch where the Fees is less than 10000.
19. Rename Sname to Fullname for students with Rollno in [106, 107, 108].
20. Rename Fees to Tuition_Fees for all students with Fees greater than 9000.
21. Rename Department to Major where the Fees is less than 15000 and Gender is 'Female'.
22. Rename City to Hometown for all students whose SEM is 3 and Department is not 'Mechanical'.

Part – C

1. Create a capped collection named" logs" with a maximum size of 100 KB and a maximum of 10 documents.

2. Insert below 12 log entries into the "logs" collection. Each entry should contain a message, level (e.g., "info", "warning", "error"), and a timestamp field. Use the insertMany() method.
   { message: "System started", level: "info", timestamp: new Date() }
   { message: "Disk space low", level: "warning", timestamp: new Date() }
   { message: "User login", level: "info", timestamp: new Date() }
   { message: "System reboot", level: "info", timestamp: new Date() }
   { message: "Error in module", level: "error", timestamp: new Date() }
   { message: "Memory usage high", level: "warning", timestamp: new Date() }
   { message: "User logout", level: "info", timestamp: new Date() }
   { message: "File uploaded", level: "info", timestamp: new Date() }
   { message: "Network error", level: "error", timestamp: new Date() }
   { message: "Backup completed", level: "info", timestamp: new Date() }
   { message: "Database error", level: "error", timestamp: new Date() }
   { message: "Service started", level: "info", timestamp: new Date() }
3. Perform find method on "logs" collection to ensure only the **last 10 documents** are retained (even though you inserted 12).
4. Insert below 5 more documents and check if the oldest ones are automatically removed.
   { message: "New log entry 1", level: "info", timestamp: new Date() }
   { message: "New log entry 2", level: "info", timestamp: new Date() }
   { message: "New log entry 3", level: "info", timestamp: new Date() }
   { message: "New log entry 4", level: "warning", timestamp: new Date() }
   { message: "New log entry 5", level: "error", timestamp: new Date() }

Lab-11 — **regex method**

Mongo DB

Perform the following queries using Regex:

Part – A Create collection named "Employee" and insert following 10 documents into it.

employee

EID	ENAME	GENDER	JOININGDATE	SALARY	CITY
1	Nick	Male	01-JAN-13	4000	London
2	Julian	Female	01-OCT-14	3000	New York
3	Roy	Male	01-JUN-16	3500	London
4	Tom	Male	NULL	4500	London
5	Jerry	Male	01-FEB-13	2800	Sydney
6	Philip	Male	01-JAN-15	7000	New York
7	Sara	Female	01-AUG-17	4800	Sydney
8	Emily	Female	01-JAN-15	5500	New York
9	Michael	Male	NULL	6500	London
10	John	Male	01-JAN-15	8800	London

1. Find employees whose name start with E.
2. Find employees whose name ends with n.
3. Find employees whose name starts with S or M in your collection.
4. Find employees where city starts with A to M in your collection.
5. Find employees where city name ends in 'ney'.
6. Display employee info whose name contains n. (Both uppercase(N) and lowercase(n))

7. Display employee info whose name starts with E and having 5 characters.
8. Display employee whose name start with S and ends in a.
9. Display EID, ENAME, CITY and SALARY whose name starts with 'Phi'.
10. Display ENAME, JOININGDATE and CITY whose city contains 'dne' as three letters in city name.
11. Display ENAME, JOININGDATE and CITY who does not belongs to city London or Sydney.
12. Find employees whose names start with 'J'.
13. Find employees whose names end with 'y'.
14. Find employees whose names contain the letter 'a'.
15. Find employees whose names contain either 'a' or 'e'.
16. Find employees whose names start with 'J' and end with 'n'.
17. Find employees whose CITY starts with 'New'.
18. Find employees whose CITY does not start with 'L'
19. Find employees whose CITY contains the word 'York'.
20. Find employees whose names have two consecutive vowels (a, e, i, o, u).
21. Find employees whose names have three or more letters.
22. Find employees whose names have exactly 4 letters.
23. Find employees whose names start with either 'S' or 'M'.
24. Find employees whose names contain 'il' anywhere.
25. Find employees whose names do not contain 'a'.
26. Find employees whose names contain any digit.
27. Find employees whose names contain exactly one vowel.
28. Find employees whose names start with any uppercase letter followed by any lowercase letter.

Part – B Create collection named "Student" and insert following 10 documents into it.

Student						
ROLLNO	**SNAME**	**DEPARTMENT**	**FEES**	**SEM**	**GENDER**	**CITY**
101	Vina	CE	15000	3	Female	Rajkot
102	Krisha	EC	8000	5	Female	Ahmedabad
103	Priti	Civil	12000	7	Female	Baroda
104	Mitul	CE	15000	3	Male	Rajkot
105	Keshav	CE	15000	3	Male	Jamnagar
106	Zarna	Civil	12000	5	Female	Ahmedabad
107	Nima	EE	9000	5	Female	Rajkot
108	Dhruv	Mechanical	10000	5	Male	Rajkot
109	Krish	Mechanical	10000	7	Male	Baroda
110	Zeel	EE	9000	3	Female	Jamnagar

1. Display documents where sname start with K.
2. Display documents where sname starts with Z or D.
3. Display documents where city starts with A to R.
4. Display students' info whose name start with P and ends with i.
5. Display students' info whose department name starts with 'C'.
6. Display name, sem, fees, and department whose city contains 'med' as three letters somewhere in city name.
7. Display name, sem, fees, and department who does not belongs to city Rajkot or Baroda.
8. Find students whose names start with 'K' and are followed by any character.
9. Find students whose names end with 'a'.
10. Find students whose names contain 'ri'. (case-insensitive)

Part – C

1. Find students whose names start with a vowel (A, E, I, O, U).
2. Find students whose CITY ends with 'pur' or 'bad'.
3. Find students whose FEES starts with '1'.
4. Find students whose SNAME starts with 'K' or 'V'.
5. Find students whose CITY contains exactly five characters.
6. Find students whose names do not contain the letter 'e'.
7. Find students whose CITY starts with 'Ra' and ends with 'ot'.
8. Find students whose names contain exactly one vowel.
9. Find students whose names start and end with the same letter.
10. Find students whose DEPARTMENT starts with either 'C' or 'E'.
11. Find students whose SNAME has exactly 5 characters.
12. Find students whose GENDER is Female and CITY starts with 'A'.

Lab-12 **Aggregate method**

Mongo DB

Perform the following queries using Aggregate:

Part – A Create collection named "Student" and insert following 10 documents into it.

Student

ROLLNO	SNAME	DEPARTMENT	FEES	SEM	GENDER	CITY
101	Vina	CE	15000	3	Female	Rajkot
102	Krisha	EC	8000	5	Female	Ahmedabad
103	Priti	Civil	12000	7	Female	Baroda
104	Mitul	CE	15000	3	Male	Rajkot
105	Keshav	CE	15000	3	Male	Jamnagar
106	Zarna	Civil	12000	5	Female	Ahmedabad
107	Nima	EE	9000	5	Female	Rajkot
108	Dhruv	Mechanical	10000	5	Male	Rajkot
109	Krish	Mechanical	10000	7	Male	Baroda
110	Zeel	EE	9000	3	Female	Jamnagar

1. Display distinct city.
2. Display city wise count of number of students.
3. Display sum of fees in your collection.
4. Display average of fees in your document.
5. Display maximum and minimum fees of your document.
6. Display city wise total fees in your collection.
7. Display gender wise maximum fees in your collection.
8. Display city wise maximum and minimum fees.
9. Display count of persons lives in Baroda city in your collection.
10. Display average fees of Rajkot city.
11. Count the number of male and female students in each Department
12. Find the total Fees collected from each Department.
13. Find the minimum Fees paid by male and female students in each City.
14. Sort students by Fees in descending order and return the top 5.

15. Group students by City and calculate the average Fees for each city, only including cities with more than 1 student.
16. Filter students from CE or Mechanical department, then calculate the total Fees.
17. Count the number of male and female students in each Department.
18. Filter students from Rajkot, then group by Department and find the average Fees for each department.
19. Group by Sem and calculate both the total and average Fees, then sort by total fees in descending order.
20. Find the top 3 cities with the highest total Fees collected by summing up all students' fees in those cities.

Part – B

1. Create a collection named" Stock."
2. Insert below 9 documents into the "Stock" collection.

```
{  "_id": 1,
   "company": "Company-A",
   "sector": "Technology",
   "eps": 5.2,
   "pe": 15.3,
   "roe": 12.8,
   "sales": 300000,
   "profit": 25000
}
{   "_id": 2,
   "company": "Company-B",
   "sector": "Finance",
   "eps": 7.1,
   "pe": 12.4,
   "roe": 10.9,
   "sales": 500000,
   "profit": 55000
}
{   "_id": 3,
   "company": "Company-C",
   "sector": "Retail",
   "eps": 3.8,
   "pe": 22.1,
   "roe": 9.5,
   "sales": 200000,
   "profit": 15000
}
{   "_id": 4,
   "company": "Company-D",
   "sector": "Technology",
   "eps": 5.2,
   "pe": 15.3,
   "roe": 12.8,
   "sales": 300000,
   "profit": 25000
}
{   "_id": 5,
```

"company": "Company-E",
"sector": "Finance",
"eps": 7.1,
"pe": 12.4,
"roe": 10.9,
"sales": 450000,
"profit": 40000
}
{ "_id": 6,
"company": "Company-F",
"sector": "Healthcare",
"eps": 3.8,
"pe": 18.9,
"roe": 9.5,
"sales": 500000,
"profit": 35000
}
{ "_id": 7,
"company": "Company-G",
"sector": "Retail",
"eps": 4.3,
"pe": 22.1,
"roe": 14.2,
"sales": 600000,
"profit": 45000
}
{
"_id": 8,
"company": "Company-H",
"sector": "Energy",
"eps": 6.5,
"pe": 10.5,
"roe": 16.4,
"sales": 550000,
"profit": 50000
}
{
"_id": 9,
"company": "Company-I",
"sector": "Consumer Goods",
"eps": 2.9,
"pe": 25.3,
"roe": 7.8,
"sales": 350000,
"profit": 20000
}

3. Calculate the total sales of all companies.

	4. Find the average profit for companies in each sector. 5. Get the count of companies in each sector/ 6. Find the company with the highest PE ratio. 7. Filter companies with PE ratio greater than 20.(Use: Aggregate) 8. Calculate the total profit of companies with sales greater than 250,000. 9. Project only the company name and profit fields.(Use: Aggregate) 10. Find companies where EPS is greater than the average EPS. 11. Group companies by sector and get the maximum sales in each sector. 12. Calculate the total sales and total profit of companies in each sector. 13. Sort companies by profit in descending order.(Use: Aggregate) 14. Find the average ROE across all companies. 15. Group companies by sector and calculate both the minimum and maximum EPS. **Part – C** 1. Count the number of companies with profit greater than 30,000. 2. Get the total profit by sector and sort by descending total profit. 3. Find the top 3 companies with the highest sales. 4. Calculate the average PE ratio of companies grouped by sector. 5. Get the sum of sales and profit for each company. 6. Find companies with sales less than 400,000 and sort them by sales. 7. Group companies by sector and find the total number of companies in each sector. 8. Get the average ROE for companies with sales greater than 200,000. 9. Find the maximum profit in each sector. 10. Get the total sales and count of companies in each sector. 11. Project fields where profit is more than 20,000 and only show company and profit. 12. Find companies with the lowest ROE and sort them in ascending order.(Use: Aggregate)

Lab-13	**Index, Cursor and Schema Validation**
Mongo DB	**Perform the following queries on Index, Cursor, Schema Validation, Embedded and Multivalued Documents:** **Part – A (Use collection "Stock" created in Lab-12)** 1. Create an index on the company field in the stocks collection. 2. Create a compound index on the sector and sales fields in the stocks collection. 3. List all the indexes created on the stocks collection. 4. Drop an existing index on the company field from the stocks collection. 5. Use a cursor to retrieve and iterate over documents in the stocks collection, displaying each document. 6. Limit the number of documents returned by a cursor to the first 3 documents in the stocks collection. 7. Sort the documents returned by a cursor in descending order based on the sales field. 8. Skip the first 2 documents in the result set and return the next documents using the cursor. 9. Convert the cursor to an array and return all documents from the stocks collection. 10. Create a collection named "Companies" with schema validation to ensure that each document must contains a company field (string) and a sector field (string). **Part – B** 1. Create a collection named "Scripts" with validation for fields like eps, pe, and roe to ensure that they are numbers and required/compulsory fields.

	2. Create a collection named "Products" where each product has an embedded document for manufacturer details and a multivalued field for categories that stores an array of category names the product belongs to. • manufacturer details: The manufacturer will be an embedded document with fields like name, country, and establishedYear. • categories: The categories will be an array field that holds multiple values. (i.e. Electronics, Mobile, Smart Devices). **Part – C** 1. Create a collection named “financial_Reports” that requires revenue (a positive number) but allows optional fields like expenses and netIncome (if provided, they should also be numbers). 2. Create a collection named "Student" where each student has name and address are embedded document and mobilenumber and emailaddress are multivalued field that stores an array of values.