Creating a new database

```
CREATE DATABASE SUST_AUTO_RICKSHAW_MANGEMENT_SYSTEM;
```

Showing already created databases

```
SHOW DATABASE;
```

Delete a database

```
DROP DATABASE SUST_AUTO_RICKSHAW_MANGEMENT_SYSTEM;
```

Creating a new database

```
CREATE DATABASE SUST_AUTO_RICKSHAW_MANGEMENT_SYSTEM
```

Creating a student table witdata and data types

```
CREATE TABLE Student(

    Roll int(5),
    Name varchar(20),
    Gender varchar(15),
    GPA double(3,2),   → numenic(3.2)
    City varchar(10),
    DateBirth Date,
    PRIMARY KEY(Roll)

);
```

Renaming Table

```
RENAME TABLE Student TO StudentInfo ;
```

Deleting Table

```
DROP TABLE StudentInfo ;
```

Insert every student whose tot_cred is greater than 100 as in instructor in the same dept with a salary of 10.000

```
INSERT INTO Instructor
SELECT ID, Name, deptName
        , 10000
FROM student
WHERR tot_cred > 100
```

Inserting Data in table

```
INSERT INTO studentinfo
VALUES
(102,'Mridul','Male',3.95,'Dhaka','2000-04-11'),
(103,'Mridul','Male',3.95,'Dhaka','2000-03-11');
```

```
INSERT INTO studentinfo
(Roll,Name,Gender,GPA,City,Datebirth)
VALUES
(102,'Mridul','Male',3.95,'Dhaka','2000-04-11'),
(103,'Mridul','Male',3.95,'Dhaka','2000-03-11');
```

Seleting Data in table (Finding Data)

Selecting Only One Column

```
SELECT Roll FROM studentinfo;
```

SELECT
FROM
WHERE

Selecting Multiple Column

```
SELECT Roll,Name
FROM studentinfo;
```

Selecting All Column

```
SELECT *
FROM studentinfo;
```

Use Distinct to remove duplicates values from selected column

DISTINCT

```
→ SELECT DISTINCT Name
  FROM studentinfo;
```

Use Order by to sort duplicates values from selected column

ORDER BY

```
  SELECT GPA
  FROM studentinfo
→ ORDER BY GPA;
```

```
SELECT GPA
FROM studentinfo
ORDER BY GPA DESC;
ORDER BY GPA DESC, Name ASC; (multiple Sorting)
```

Use Where to Search Data from selected column with condition

WHERE

```
SELECT Roll
FROM studentinfo
WHERE GPA > 3.90 ;
```

BETWEEN

→ where salary between 9000 and 10000

OR where salary <= 10.000 and salary >= 9000

```
SELECT GPA
FROM studentinfo
ORDER BY GPA DESC;
```

Updating Data in table

```
UPDATE studentinfo
SET Name = 'Sumonta'
WHERE Roll = 102;
```

Deleting Data in table

```
DELETE FROM studentinfo
WHERE Roll = 103;
```

Add Foreign Key

```
// The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.
// The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
);
```

AND, OR and NOT Operators

```
SELECT * FROM Customers
WHERE Country = 'Germany' AND City = 'Berlin';


// if any of the conditions separated by OR is TRUE.

SELECT * FROM Customers
WHERE City = 'Berlin' OR City = 'Stuttgart';
```

Instructor whose salary is above 10,000 recive a 3% raise in salarc

```
update instructor        3%↑
set salary = salary * 1.03
where salary > 10000
```

Delete all course that have never been offered (do not occur in section relation)

```
DELETE from course
where course_ID NOTIN
      ( SELECT course_ID
        FROM section);
```

Joining table

```
SELECT orders.OrdrID, Customers.CustomerName,
                  orders.OrderDate


FROM Orders
INNER JOIN    customers ON orders.CustomerID
                        = customers.CustomerID;
```

Delete all year-2010 cars belonging to the person whose ID is '12345'

Sol-1

```
DELECT FROM car
WHERE year = 2010
and licesnce.plate in
(SELECT license-plate
 FROM owns
 WHERE owns.driverID
          = '12345)
```

```
// if the condition(s) is NOT TRUE.

SELECT * FROM Customers
WHERE NOT Country = 'Germany';
```

CHECK on CREATE TABLE

```
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
CHECK (Age>=18)
```

Consider the following Banking Database

- branch(branch name, branch city, assets)
- customer(customer name, customer street, customer city)
- loan(loan number, brach name, amount)
- borrower(customer name, loan number)
- account(account number, branch name, balance)
- depositor(customer name, account number)

1 Find the names of all branched located in "Dhaka"

```
SELECT branch_Name
FROM branch
WHERE branch_city = "Dhaka";
```

2 Find the names of all borrowers who have a loan in branch "Mirpur"

```
SELECT customer_name
FROM Borrower, Loan
WHERE Borrower.loan_number = loan.loan_number
and branch_name = "Mirpur";
```

Renaming Relation using "as"

```
SELECT B.customer.name
FROM Borrower as B,
     loas as L
WHERE
  B.loanNumber
     = L.load Number
```

3 Find all loan numbers with a loan value greater than BDT100,000

```
SELECT loan_number
FROM loan
WHERE amount > 100000;
```

4 Find the names of all depositors who have an account with a value greater than BDT60,000

```
SELECT customer_name
FROM depositor, account
WHERE account.account_number = depositor.account_number
and balance > 60000;
```

5 Find the names of all depositors who have an account with a value greater than BDT60,000 at the "Motejheel" branch

```
SELECT customer_name
FROM depositor, account
WHERE account.account_number = depositor.account_number
end balance > 60000
and branch_city = "Motejheel";
```

Q-3.8

6 FIND the ID of each customer of the bank who has an account but not a loan

```
(SELECT   ID
 FROM     Depositor)
except
(SELECT ID
 FROM borrower
          );
```

7 Find the ID of each customer who lives on same street and in the same city as customer '12345'

```
SELECT   ID
FROM     customer as F ,  customer   as D
WHERE    F.city = D.city   and
         F.street = D.street and
         D. ID  =  '12345'
```

Renaming columns :

```
ALTER TABLE tablename CHANGE oldcolname
newcolname datatype (length);
```

- . -

NOT equal <>

Q find the ID of each employee who doesn't work for 'ABC'

```
SELECT ID
FROM works
WHERE company name <> 'ABC
```

another way

```
SELECT ID
FROM employee

where ID Not in

( SELECT ID
  FROM works
  WHERE company.name ='ABC'

                  );
```

Creating a table

Roll	Name	Gender	GPA	City	Date of Birth

```
CREATE TABLE TableName
(

columnName1 datatype(size),

columeName2 datatype(size),

columnName3 datatype(size)

);
```

varchar(20)

- maximum 20 char
- if use 5char it will use 5char not whole 20

~~create~~
```
CREATE TABLE student
(
 Roll  int (5),
 Name  Varchar(20);
 Gender  varchar (10),
 GPA  double (3, 2),
 city  varchar(15),
 Date-Birth  Date

 PRIMARY ~~KEY~~ KEY(Roll)
```

① Find the names of all branch located in "Dhaka"

```
→ SELECT   Branch_Name
  FROM     branch
  WHERE    branch_city = 'Dhaka' ;
```

② find the names of all borrower who have a loan in branch "Mirpur"

```
SELECT   customer-Name
FROM     borrower, loan
WHERE    borrower. loan-Number
              = loan. loan-Number
         and branch-Name = "Dhaka"
```

(iii) find all loan number with a loan value greater BDT 1000.00

SELECT ~~amount~~ loan number

FROM loan

WhERE amont > 100000 ;

(iv) name all depositors who have an account with a value greater than 60.000

SELECT customer_name

FROM depositor, account

where depositor.account_number
= account. account_number

and ~~amoun~~ balance > 60000 ;

(v) ·Find the names of all depoistors who have an account with a greater value than 60,000 at motijeel branch.

```
SELECT   customer_Name
FROM     Account, Depositor
where    Depositor. account_Number
              = Account. account_Number
         and branch_name = 'Motijhee'
          and balance >60,000
```

Basic Types

char(n) : fixed length character string

varchar(n) : Variable length " "

char(8) vs varchar(8)

→ fixed 8 size → any from size (1-8)

int , int(n), float (n)

numeric (p. d)

Ex:- numeric (3, 1)

allows → $\underset{3}{\underline{44.\overset{1}{3}}}$ type only

NOT NULL → will not accept null values

Types-1 :- while creating table

```
CREATE TABLE Persons
(
    ID int NOT NULL
);
```

Type-2 :- after creating table

```
ALTER TABLE Person
ALTER COLUMN ID int NOT NULL;
```

show result if we gave a 10% raise to each instructor.

```
SELECT salary * 1.1
FROM instructor;
```

→ 10% raise

String Matching

'%' → match any substring

'_' → match any string

'Intro%' → match any string begin with "Intro"

'%gg%' → match any string that contains gg

'%-03-%' → match any date that includes only the month march

'___' → match any string that has exactly 3 character

'__%' → match any string at least three character

To match %, _ use '\'

'\%' → % match

'_' → _ match

'\\' → \ match

#Check Instructor who 'Saha' in their Name

```
SELECT Name
FROM Instructor
WHERE name like '%Saha%'
```

checking 'saha'

NT :- LIKE case sensitive

String lower conversion

lower (name)

Union Operation 'U'

Q Find all course taught $\underset{U}{\text{either}}$ in Fall 2017 or Spring 2018

```
(SELECT course.ID
 FROM semester
 where semester.name = 'Fall' and
          Year = '2017')

union

(SELECT course.ID
 FROM semester
 WHERE   semester.name = 'Spring' and
          year = '2018');
```

Union → auto remove duplicates values

Union all → allows duplicates

Intensect ∩

Q → Find all course taught in both ∩ the Fall 2017 and spring 2018

(   )

intensect

(   );

intensect all
→ dupli values

Except

Q Find all course taught in the Fall 2017 but not in spring 2018

(   )

except

(   );

except all

Find average salary of instructor

```
SELECT avg(salary)
FROM instructor ;
```

Gives average salary

Do it wth in more meaning full way

```
SELECT avg(salary) as avg-salary
FROM instructor;
```

new attribute

Like → avg , sum , min, max

count the number of teacher who teaches in 2018

```
SELECT count (distinct Name)
FROM teacher
where year = '2018'
```

counting

Grouping

group dept

Q find average salary in each department

```
SELECT  deptName, avg(salary) as avg_sal
FROM  instructor
group by  deptName ;
```

grouping dept name wise

—— o ——

SubQuery

Find teacher whose salary is greater than average of all teacher

```
SELECT  name
FROM  Teacher
where salary > ( select avg(salary)
                 FROM Teacher);
```

Sub Queny — using In

Q/ find all the counse taught in both fall 2017 and spning 2018

```
SFLFCT Distinct  counse_name
fROM SECTION
WHERR  semesten  = 'spning'  and yean = '2018'

      and counse-name in (

           SELECT  counse-name
           FROM  SECTION
           WHERE  Semesten = 'fall'
                       and yean = '2017'

       );
```

→ subqueny

—·—

multiple selections

```
# whene (Name, Roll) in (select Name, Roll
                    .......
                          );
```

Q 3.5b assign grades to students based on the score as follow, if score <40 grade f., greade c if $40 \le$ score <60, B if $60 \le$ score <80, A if score ≥ 80

Q → Find the Number of students with each grade

```
SELECT count(ID),
        case
         when   score <40    then 'F'
         when   score < 60   then 'C'
         when   score < 80   then 'B'
         else 'A'
        end as grade

FROM ~~GRADES~~ Marks
GROUP BY   grade
```

3-9

Q Find ID of each employee who earn more than every employee of 'A Bank'

Sol-1

```
SELECT  ID
FROM  employee, works
WHERE   salary  > all (
                 select salary
                 FROM works
                 WHERE company-name = 'ABank'
                          );
```

Q Find the name of Company that has most employees

Sol

```
SELECT    company-name
FROM      works
~~SEO~~ Group By  company- name
      Having    count (distinct ID) > all

                ( select  count(distinct ID)
                  FROM     works
                  GROUP by   company Name);
```