**1 a) create sailors, boats, reserves tables using the following fields.**

**Note: sid is the primary key for sailors table**

**bid is the primary key for boats table**

**sid,bid is foreign keys in reserves table**

**Sailors(sid,sname,rating,age);**

**Boats(bid,bname,color);**

**Reserves(sid,bid,day);**

CREATE TABLE sailors (

-> sid integer not null,

-> sname varchar(32),

-> rating integer,

-> age real,

-> CONSTRAINT PK\_sailors PRIMARY KEY (sid)

-> );

Query OK, 0 rows affected (0.03 sec)

CREATE TABLE boats (

-> bid integer not null,

-> bname varchar(25),

-> color varchar(21),

-> CONSTRAINT PK\_boats PRIMARY KEY (bid)

-> );

Query OK, 0 rows affected (0.02 sec)

CREATE TABLE reserves (

-> sid integer not null,

-> bid integer not null,

-> day datetime not null,

-> CONSTRAINT PK\_reserves PRIMARY KEY (sid, bid, day),

-> FOREIGN KEY (sid) REFERENCES sailors(sid),

-> FOREIGN KEY (bid) REFERENCES boats(bid)

-> );

mysql> INSERT INTO sailors (sid, sname, rating, age) VALUES (22, 'Dustin', 7, 45.0), (23, 'John', 8, 35.0), (24, 'Mike', 9, 25.0);

Query OK, 3 rows affected (0.02 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> INSERT INTO boats (bid, bname, color) VALUES (101, 'Boat1', 'red'), (102, 'Boat2', 'green'), (103, 'Boat3', 'blue');

Query OK, 3 rows affected (0.01 sec)

Records: 3 Duplicates: 0 Warnings: 0

mysql> INSERT INTO reserves (sid, bid, day) VALUES (22, 101, '1998-10-10'), (23, 102, '1998-10-11'), (24, 103, '1998-10-12');

Query OK, 3 rows affected (0.01 sec)

Records: 3 Duplicates: 0 Warnings: 0

**b) Write a SQL Query to find name of the sailors who have reserved both red or green**

**boat**

mysql> SELECT s.sname

-> FROM sailors s

-> JOIN reserves r ON r.sid=s.sid

-> JOIN boats b ON r.bid=b.bid AND b.color='red'

-> WHERE r.sid IN(SELECT s.sid FROM sailors s JOIN reserves r ON r.sid=s.sid JOIN boats b ON r.bid=b.bid WHERE b.color='green');

Empty set (0.00 sec)

**c) Write a SQL Query to find the name of the sailors who have reserved boat bid 103**

mysql> SELECT S.sname

-> FROM Sailors S

-> WHERE EXISTS (SELECT \* FROM Reserves R WHERE R.bid=103 AND R.sid=S.sid);

+-------+

| sname |

+-------+

| Mike |

+-------+

1 row in set (0.01 sec)

**d) Write a SQL Query to find the name of the sailors who have not reserved boat bid 104**

mysql> SELECT s.sname

-> FROM sailors s

-> WHERE NOT EXISTS (

-> SELECT \*

-> FROM reserves r

-> WHERE r.sid = s.sid AND r.bid = 104

-> );

+--------+

| sname |

+--------+

| Dustin |

| John |

| Mike |

+--------+

3 rows in set (0.00 sec)

**e) Write a SQL Query to find all SIDs of sailors who have a rating of 10 or reserved boat 104**

mysql> SELECT DISTINCT r.sid

-> FROM reserves r

-> JOIN sailors s ON r.sid = s.sid

-> WHERE s.rating = 10 OR r.bid = 104;

Empty set (0.01 sec)

**2 a) create sailors, boats, reserves tables using the following fields.**

**Note: sid is the primary key for sailors table**

**bid is the primary key for boats table**

**sid,bid is foreign keys in reserves table**

**Sailors(sid,sname,rating,age);**

**Boats(bid,bname,color);**

**Reserves(sid,bid,day);**

CREATE TABLE sailors (

sid INT PRIMARY KEY,

sname VARCHAR(255),

rating INT,

age INT

);

CREATE TABLE boats (

bid INT PRIMARY KEY,

bname VARCHAR(255),

color VARCHAR(255)

);

CREATE TABLE reserves (

sid INT,

bid INT,

day DATE,

PRIMARY KEY (sid, bid),

FOREIGN KEY (sid) REFERENCES sailors(sid),

FOREIGN KEY (bid) REFERENCES boats(bid)

);

**b) Write a SQL Query to display average age of all sailors**

mysql> SELECT AVG(age) AS avg\_age

-> FROM sailors;

+---------+

| avg\_age |

+---------+

| 35 |

+---------+

1 row in set (0.01 sec)

**c) Write a SQL Query to display number of sailors in the sailors table**

mysql> SELECT COUNT(\*) AS num\_sailors

-> FROM sailors;

+-------------+

| num\_sailors |

+-------------+

| 3 |

+-------------+

1 row in set (0.02 sec)

**d) Write a SQL Query to display name of the sailor who is older than all sailors**

mysql> SELECT sname

-> FROM sailors

-> WHERE age = (

-> SELECT MAX(age)

-> FROM sailors

-> );

+--------+

| sname |

+--------+

| Dustin |

+--------+

1 row in set (0.00 sec)

**e) Write a SQL Query to find the average age of sailors who are of voting age (i.e., at least 18 years old) for each rating level that has at least two sailors.**

mysql> SELECT rating, AVG(age) AS avg\_age

-> FROM sailors

-> WHERE age >= 18

-> GROUP BY rating

-> HAVING COUNT(\*) >= 2;

Empty set (0.00 sec)

**3. a) create sailors, boats, reserves tables using the following fields.**

**Note: sid is the primary key for sailors table**

**bid is the primary key for boats table**

**sid,bid is foreign keys in reserves table**

**Sailors(sid,sname,rating,age);**

**Boats(bid,bname,color);**

**Reserves(sid,bid,day);**

Same as 1(a)

**b) Write a SQL query to create a view with the name young\_sailors (age <30)**

**Young\_sailors(sid,sname,age)**

mysql> CREATE VIEW young\_sailors AS

-> SELECT sid, sname, age

-> FROM sailors

-> WHERE age < 30;

Query OK, 0 rows affected (0.01 sec)

**c) Write a SQL query to create a view with the name old\_sailors (age >30)**

**old\_sailors(sid,sname,age)**

mysql> CREATE VIEW old\_sailors AS

-> SELECT sid, sname, age

-> FROM sailors

-> WHERE age > 30;

Query OK, 0 rows affected (0.01 sec)

**d) Write a SQL query to delete young\_sailors view.**

mysql> DROP VIEW IF EXISTS young\_sailors;

Query OK, 0 rows affected (0.01 sec)

mysql> select \*from young\_sailors;

ERROR 1146 (42S02): Table 'bsr.young\_sailors' doesn't exist

**e) Write a SQL query to delete old\_sailors view.**

mysql> select \*from old\_sailors;

+-----+--------+------+

| sid | sname | age |

+-----+--------+------+

| 22 | Dustin | 45 |

| 23 | John | 35 |

+-----+--------+------+

2 rows in set (0.01 sec)

mysql> DROP VIEW IF EXISTS old\_sailors;

Query OK, 0 rows affected (0.01 sec)

mysql> select \*from old\_sailors;

ERROR 1146 (42S02): Table 'bsr.old\_sailors' doesn't exist

**4. a) create sailors, boats, reserves tables using the following fields.**

**Note: sid is the primary key for sailors table**

**bid is the primary key for boats table**

**sid,bid is foreign keys in reserves table**

**Sailors(sid,sname,rating,age);**

**Boats(bid,bname,color);**

**Reserves(sid,bid,day);**

Same as 1(a)

**b) What are triggers? Explain with detailed syntax for creating triggers?**

In SQL, a trigger is a special type of stored procedure that is automatically executed in response to certain events or changes in the database. These events can include INSERT, UPDATE, and DELETE operations on a table. Triggers are used to enforce business rules, perform complex calculations, and maintain data integrity.

example of the syntax for creating a trigger:

CREATE TRIGGER trigger\_name

AFTER INSERT ON table\_name

FOR EACH ROW

BEGIN

-- trigger code goes here

END;

**c) Demonstrate creating trigger with an example**

mysql> delimiter //

mysql> create trigger tr1 before insert on sailors

-> for each row

-> begin

-> if new.age>50 then set new.age=50;

-> else

-> set new.age=40;

-> end if;

-> end;

-> //

Query OK, 0 rows affected (0.02 sec)

mysql> insert into sailors values(25,'joe',5,55);

-> //

Query OK, 1 row affected (0.01 sec)

mysql> select \*from sailors;

-> //

+-----+--------+--------+------+

| sid | sname | rating | age |

+-----+--------+--------+------+

| 22 | Dustin | 7 | 45 |

| 23 | John | 8 | 35 |

| 24 | Mike | 9 | 25 |

| 25 | joe | 5 | 50 |

+-----+--------+--------+------+

4 rows in set (0.00 sec)

**5. a) create sailors, boats, reserves tables using the following fields.**

**Note: sid is the primary key for sailors table**

**bid is the primary key for boats table**

**sid,bid is foreign keys in reserves table**

**Sailors(sid,sname,rating,age);**

**Boats(bid,bname,color);**

**Reserves(sid,bid,day);**

Same as 1(a)

**b) Write a procedure to retrieve name and age of sailors whose age is more than 30**

mysql> CREATE PROCEDURE pr(age int)

-> BEGIN

-> SELECT name, age

-> FROM sailors

-> WHERE age > 30;

-> END;

-> //

Query OK, 0 rows affected (0.04 sec)

**c) Write a procedure to display names of the boat with red color.**

mysql> CREATE PROCEDURE p2(color int)

-> BEGIN

-> SELECT name

-> FROM boat

-> WHERE color = 'red';

-> END;

-> //

Query OK, 0 rows affected (0.01 sec)

**d) Write a procedure to display names of the sailors who reserved green color boat.**

mysql> CREATE PROCEDURE p3(color int)

-> BEGIN

-> SELECT sailors.name

-> FROM sailors

-> JOIN reserves ON sailors.sid = reserves.sid

-> JOIN boat ON reserves.bid = boat.bid

-> WHERE boat.color = 'green';

-> END;

-> //

Query OK, 0 rows affected (0.02 sec)

**6. What is the need of normalization? Explain 1st, 2nd and 3rd normal forms with suitable examples?**

Normalization is an important process in database design that helps in improving the efficiency, consistency, and accuracy of the database. [It makes it easier to manage and maintain the data and ensures that the database is adaptable to changing business needs1](https://www.geeksforgeeks.org/introduction-of-database-normalization/).

The need for normalization arises due to several reasons. Some of them are:

1. It eliminates redundant data.
2. It reduces chances of data error.
3. The normalization is important because it allows the database to take up less disk space.
4. It also helps in increasing performance.
5. [It improves the data integrity and consistency2](https://www.geeksforgeeks.org/what-is-data-normalization-and-why-is-it-important/).

Normalization is divided into several normal forms such as 1st normal form (1NF), 2nd normal form (2NF), 3rd normal form (3NF), and Boyce-Codd normal form (BCNF). Each normal form has its own set of rules that must be followed to ensure that the database is normalized properly.

[First Normal Form (1NF) - A relation is said to be in 1NF if it contains no repeating groups or arrays3](https://hackr.io/blog/dbms-normalization).

[Second Normal Form (2NF) - A relation is said to be in 2NF if it is in 1NF and every non-key attribute is fully dependent on the primary key3](https://hackr.io/blog/dbms-normalization).

[Third Normal Form (3NF) - A relation is said to be in 3NF if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key3](https://hackr.io/blog/dbms-normalization).

For example, consider a table named “Student” with columns “Student ID”, “Student Name”, “Course ID”, “Course Name”, “Course Instructor”. This table violates the first normal form because it contains repeating groups of data such as “Course ID”, “Course Name”, and “Course Instructor”. To normalize this table, we can create two tables named “Student” and “Course” with columns as shown below:

**7. What is normalization? Explain BCNF, 4th and 5th normal forms with suitable examples?**

Normalization is an important process in database design that helps in improving the efficiency, consistency, and accuracy of the database. [It makes it easier to manage and maintain the data and ensures that the database is adaptable to changing business needs1](https://www.geeksforgeeks.org/normal-forms-in-dbms/).

Normalization is divided into several normal forms such as 1st normal form (1NF), 2nd normal form (2NF), 3rd normal form (3NF), Boyce-Codd normal form (BCNF), 4th normal form (4NF), and 5th normal form (5NF). Each normal form has its own set of rules that must be followed to ensure that the database is normalized properly.

Boyce-Codd Normal Form (BCNF) - A relation is said to be in BCNF if it is in 3NF and for every functional dependency X → Y, X must be a superkey[2](https://www.javatpoint.com/dbms-normalization).

Fourth Normal Form (4NF) - A relation is said to be in 4NF if it is in BCNF and has no non-trivial multi-valued dependencies[3](https://www.guru99.com/database-normalization.html)[4](https://www.geeksforgeeks.org/introduction-of-4th-and-5th-normal-form-in-dbms/).

Fifth Normal Form (5NF) - A relation is said to be in 5NF if it is in 4NF and cannot be decomposed into any number of smaller tables without loss of data[3](https://www.guru99.com/database-normalization.html)[5](https://www.tutorialspoint.com/Fifth-Normal-Form-5NF).

For example, consider a table named “Employee” with columns “Employee ID”, “Employee Name”, “Department ID”, “Department Name”, “Project ID”, “Project Name”. This table violates the BCNF because there are functional dependencies such as “Department ID” → “Department Name” and “Project ID” → “Project Name” which do not satisfy the superkey condition. To normalize this table, we can create three tables named “Employee”, “Department”, and “Project” with columns as shown below:

Table 1: Employee

| Employee ID | Employee Name | Department ID | Project ID |
| --- | --- | --- | --- |
| 1 | John | 101 | 201 |
| 2 | Jane | 102 | 202 |

Table 2: Department

| Department ID | Department Name |
| --- | --- |
| 101 | Sales |
| 102 | Marketing |

Table 3: Project

| Project ID | Project Name |
| --- | --- |
| 201 | Project A |
| 202 | Project B |

[Here, we have eliminated the functional dependencies by creating three separate tables3](https://www.guru99.com/database-normalization.html)

**8) a) create sailors, boats, reserves tables using the following fields.**

**Note: sid is the primary key for sailors table**

**bid is the primary key for boats table**

**sid,bid is foreign keys in reserves table**

**Sailors(sid,sname,rating,age);**

**Boats(bid,bname,color);**

**Reserves(sid,bid,day);**

Same as 1(a)

**b) What are cursors? Write syntax for:**

1. **Declaring a cursor ii) Opening a cursor iii) Fetching a cursor iv) closing a cursor**

**Cursor:** Cursors are used when the SQL Select statement is expected to return more than one row. Cursors are supported inside procedures and functions. Cursors must be declared and its definition contains the query. The cursor must be defined in the DECLARE section of the program. A cursor must be opened before processing and close after processing.   
Syntax to declare the cursor:   
        **DECLARE <cursor\_name> CURSOR FOR <select\_statement>**

Multiple cursors can be declared in the procedures and functions but each cursor must have a unique name. And in defining the cursor the select\_statement cannot have INTO clause.    
Syntax to open the cursor:   
**OPEN <cursor\_name>**

By this statement we can open the previously declared cursor.   
Syntax to store data in the cursor :   
      **FETCH <cursor\_name> INTO <var1>,<var2>…….**

The above statement is used to fetch the next row if a row exists by using the defined open cursor.   
Syntax to close the cursor :   
        **CLOSE <cursor\_name>**

By this statement we can close the previously opened cursor. If it is not closed explicitly then a cursor is closed at the end of compound statement in which that was declared

mysql> create procedure kcurl1(s\_id int)

-> begin

-> declare x\_no int;

-> declare x\_name varchar(50);

-> declare x\_age int;

-> declare c5 cursor for select sid,sname,age from sailors where sid=s\_id;

-> open c5;

-> fetch c5 into x\_no,x\_name,x\_age;

-> select x\_no,x\_name,x\_age from sailors where sid=s\_id;

-> close c5;

-> end;

-> //

Query OK, 0 rows affected (0.01 sec)