CO - 2:

1.

Here are the SQL queries for the given tasks, incorporating sub-queries and joins:

(i) SQL query to display the pid and pname of the project which belongs to finance and health domain:

```sql
SELECT pid, pname
FROM project
WHERE domain IN ('Finance', 'Health');
```

(ii) SQL query to display the pid, empid, and pname of the project that does not belong to finance and health domain:

```sql
SELECT pa.pid, pa.empid, p.pname
FROM projectallocation pa
JOIN project p ON pa.pid = p.pid
WHERE p.domain NOT IN ('Finance', 'Health');
```

(iii) SQL query to display the allocationid, pid, empid from project allocation table whose allocation date is greater than June 2019 and less than June 2020:

```sql
SELECT allocationid, pid, empid
FROM projectallocation
WHERE allocationdate > '2019-06-01' AND allocationdate < '2020-06-01';
```

(iv) Display PID as NP and ALLOCATIONDATE as NA, for the employee who have DevOps skills, or for the employee who is NOT yet allocated to any project:

```sql
SELECT COALESCE(pa.pid, 'NP') AS PID, COALESCE(pa.allocationdate, 'NA') AS
ALLOCATIONDATE
FROM employee e
LEFT JOIN projectallocation pa ON e.empid = pa.empid
WHERE e.skills = 'DevOps' OR pa.pid IS NULL;
```

(v) Display EMPID and PID for the employee who does NOT have Security skills:

```sql
SELECT pa.empid, pa.pid
FROM projectallocation pa
JOIN employee e ON pa.empid = e.empid
WHERE e.skills != 'Security';
```

2.
Here are the SQL queries for the given tasks:

(i) SQL query to display the Designation without duplicates:

```sql
SELECT DISTINCT DESIGNATION
FROM Employee;
```

(ii) SQL query to display ID and ENAME of the employee whose salary is greater than the average salary of all:

```sql
SELECT ID, ENAME
FROM Employee
WHERE SALARY > (SELECT AVG(SALARY) FROM Employee);
```

(iii) Display the total count of employees in each department:

```sql
SELECT DEPT, COUNT(*) AS TotalEmployees
FROM Employee
GROUP BY DEPT;
```

(iv) SQL query to display the Total salary paid in each department:

```sql
SELECT DEPT, SUM(SALARY) AS TotalSalary
FROM Employee
GROUP BY DEPT;
```

(v) Display BONUS column as NA for those who do not have a bonus:

```sql
```

```
SELECT ID, ENAME, COALESCE(BONUS, 'NA') AS BONUS
FROM Employee;
```

3.
Here are the SQL queries for the given tasks:

(i) SQL query to display the book names taken by the member 1 (MId=1):

```sql
SELECT B.Name
FROM Book B
INNER JOIN Issue I ON B.BId = I.BId
WHERE I.MId = 1;
```

(ii) SQL query to display the book name and issue date of the member 3:

```sql
SELECT B.Name, I.IssueDate
FROM Book B
INNER JOIN Issue I ON B.BId = I.BId
WHERE I.MId = 3;
```

(iii) SQL query to display the book name and number of days the book held by each member:

```sql
SELECT B.Name, I.MId, DATEDIFF(day, I.IssueDate, I.ReturnDate) AS DaysHeld
FROM Book B
INNER JOIN Issue I ON B.BId = I.BId;
```

(iv) SQL query to display the MId, BId, BName having fine:

```sql
SELECT I.MId, I.BId, B.Name AS BName
FROM Issue I
INNER JOIN Book B ON I.BId = B.BId
WHERE I.Fine IS NOT NULL;
```

(v) SQL query to display the count of each book distributed to the members:

```sql
SELECT B.Name AS BookName, COUNT(*) AS DistributionCount
FROM Book B
```

```
INNER JOIN Issue I ON B.BId = I.BId
GROUP BY B.Name;
```

4.

Here are the SQL queries for the given tasks:

(i) SQL query to display the prdid, prdname, and cost:

```sql
SELECT PRDID, PRDNAME, COST
FROM Product;
```

(ii) SQL query to display the prdid, prdname, and total selling cost as TSC for each order id:

```sql
SELECT o.ORDERID, p.PRDID, p.PRDNAME, SUM(p.COST) AS TSC
FROM Orders o
JOIN Product p ON o.PRDID = p.PRDID
GROUP BY o.ORDERID, p.PRDID, p.PRDNAME;
```

(iii) SQL query to display the highest selling product with prdid and prdname:

```sql
SELECT TOP 1 p.PRDID, p.PRDNAME
FROM Orders o
JOIN Product p ON o.PRDID = p.PRDID
GROUP BY p.PRDID, p.PRDNAME
ORDER BY SUM(p.COST) DESC;
```

(iv) SQL query to display the prdid and prdname sold greater than the average of total sales:

```sql
SELECT p.PRDID, p.PRDNAME
FROM Orders o
JOIN Product p ON o.PRDID = p.PRDID
GROUP BY p.PRDID, p.PRDNAME
HAVING SUM(p.COST) > (SELECT AVG(TotalSales) FROM (SELECT SUM(COST) AS TotalSales
FROM Orders GROUP BY PRDID) AS SalesAvg);
```

(v) SQL query to display the count of sales of each product:

```sql
SELECT p.PRDID, p.PRDNAME, COUNT(*) AS SalesCount
FROM Orders o
JOIN Product p ON o.PRDID = p.PRDID
GROUP BY p.PRDID, p.PRDNAME;
```

[20/03, 9:43 pm] ***Meenakshi Arumugam*** ❣️: 5.

Here are the SQL queries for the given tasks:

(i) SQL query to display the empno, ename, and dname:

```sql
SELECT emp.empno, emp.ename, dept.dname
FROM emp
INNER JOIN dept ON emp.deptno = dept.deptno;
```

(ii) Display the empno, ename, and dname as NA for employees who do not belong to any department:

```sql
SELECT emp.empno, emp.ename, COALESCE(dept.dname, 'NA') AS dname
FROM emp
LEFT JOIN dept ON emp.deptno = dept.deptno;
```

(iii) SQL query to display the ename, empno, deptno whose salary is greater than the average salary of all:

```sql
SELECT ename, empno, deptno
FROM emp
WHERE sal > (SELECT AVG(sal) FROM emp);
```

(iv) SQL query to display the department-wise total Salary:

```sql
SELECT dept.dname, SUM(emp.sal) AS total_salary
FROM emp
INNER JOIN dept ON emp.deptno = dept.deptno
GROUP BY dept.dname;
```

(v) SQL query to display the ename of the 3rd highest salary:

```sql
SELECT ename
FROM (
   SELECT ename, RANK() OVER (ORDER BY sal DESC) AS salary_rank
   FROM emp
) AS ranked_emp
WHERE salary_rank = 3;
```
[20/03, 9:44 pm] *Meenakshi Arumugam* ❣️: 6.

Here are the SQL queries for the given tasks:

(i) SQL query to display the city and region for all the products purchased:

```sql
SELECT city, region
FROM store
INNER JOIN product ON store.store_id = product.store_id;
```

(ii) SQL query to display the productid and desc from the east and west region:

```sql
SELECT productid, desc
FROM product
INNER JOIN store ON product.store_id = store.store_id
WHERE region IN ('East', 'West');
```

(iii) SQL Query to display the storeid as NP that does not sell anything:

```sql
SELECT s.store_id, COALESCE(p.productid, 'NP') AS productid
FROM store s
LEFT JOIN product p ON s.store_id = p.store_id
WHERE p.productid IS NULL;
```

(iv) SQL query to display the storeid, city, region which has more sales:

```sql
SELECT store_id, city, region
FROM (
   SELECT store_id, city, region, ROW_NUMBER() OVER (ORDER BY COUNT(productid) DESC)
AS sales_rank
```

```
    FROM product
    GROUP BY store_id, city, region
) AS ranked_sales
WHERE sales_rank = 1;
```

(v) SQL query to display the count of stores based on region:

```sql
SELECT region, COUNT(DISTINCT store_id) AS store_count
FROM store
GROUP BY region;
```

[20/03, 9:45 pm] *Meenakshi Arumugam* ❣️: 7.

Here are the SQL queries for the given tasks:

(i) SQL query to display the total amount paid through each payment mode:

```sql
SELECT PymtMode, SUM(Amount) AS TotalAmountPaid
FROM Payment
GROUP BY PymtMode;
```

(ii) SQL query to display the pymtId and location of payments made after 31-Jan-15:

```sql
SELECT PymtId, Location
FROM Payment
WHERE PymtDt > '2015-01-31';
```

(iii) SQL query to display the count of each payment mode in each location:

```sql
SELECT Location, PymtMode, COUNT(*) AS PaymentCount
FROM Payment
GROUP BY Location, PymtMode;
```

(iv) SQL query to display the pymtID and pymtMode which have an amount greater than the average amount of all:

```sql
SELECT PymtId, PymtMode
```

FROM Payment
WHERE Amount > (SELECT AVG(Amount) FROM Payment);
```

(v) SQL query to display the location of the pymtId whose payment is less than 50 and the location is New York:

```sql
SELECT Location
FROM Payment
WHERE Amount < 50 AND Location = 'New York';
```
[20/03, 9:47 pm] *Meenakshi Arumugam* ❣️: 8.

Here are the SQL queries for the given tasks:

(i) SQL query to display the donorid and patientid who has:

```sql
SELECT bt.DONORID, bt.PATIENTID
FROM BloodTransaction bt
WHERE bt.DONORID IN (SELECT DONORID FROM Donor WHERE DONORNAME = 'Sam');
```

(ii) SQL query to display the donorid and donorname with the donated count:

```sql
SELECT bt.DONORID, d.DONORNAME, COUNT(bt.DONORID) AS DonatedCount
FROM BloodTransaction bt
JOIN Donor d ON bt.DONORID = d.DONORID
GROUP BY bt.DONORID, d.DONORNAME;
```

(iii) SQL query to display the transid of AB+:

```sql
SELECT TRANSID
FROM BloodTransaction
WHERE DONORID IN (SELECT DONORID FROM Donor WHERE BLOODGROUP = 'AB+');
```

(iv) SQL query to display the patientid, donorid, donorname who donated after August 2013:

```sql
SELECT bt.PATIENTID, bt.DONORID, d.DONORNAME
FROM BloodTransaction bt
```
```

JOIN Donor d ON bt.DONORID = d.DONORID
WHERE TRANSDATE > '2013-08-01';
```

(v) SQL query to display the blood group of each patient with their patientid:

```sql
SELECT PATIENTID, BLOODGROUP
FROM BloodTransaction
JOIN Donor ON BloodTransaction.DONORID = Donor.DONORID;
```
[20/03, 9:48 pm] *Meenakshi Arumugam*❣️: 9.

Here are the SQL queries for the given tasks:

(i) SQL query to display the orderid and supplier name for orders made:

```sql
SELECT o.ORDER_ID, s.SUPPLIER_NAME
FROM ORDERS o
INNER JOIN SUPPLIER s ON o.SUPPLIER_ID = s.SUPPLIER_ID;
```

(ii) SQL query to display the supplier id as NP for those who do not have any orders:

```sql
SELECT s.SUPPLIER_ID, COALESCE(o.ORDER_ID, 'NP') AS ORDER_ID
FROM SUPPLIER s
LEFT JOIN ORDERS o ON s.SUPPLIER_ID = o.SUPPLIER_ID;
```

(iii) SQL query to display the orderdate and supplier name for the orders made:

```sql
SELECT o.ORDER_DATE, s.SUPPLIER_NAME
FROM ORDERS o
INNER JOIN SUPPLIER s ON o.SUPPLIER_ID = s.SUPPLIER_ID;
```

(iv) SQL query to display the supplier name without duplication:

```sql
SELECT DISTINCT SUPPLIER_NAME
FROM SUPPLIER;
```
```

(v) SQL query to display the count of orders for each supplier id:

```sql
SELECT s.SUPPLIER_ID, COUNT(o.ORDER_ID) AS OrderCount
FROM SUPPLIER s
LEFT JOIN ORDERS o ON s.SUPPLIER_ID = o.SUPPLIER_ID
GROUP BY s.SUPPLIER_ID;
```

[20/03, 9:49 pm] *Meenakshi Arumugam* ❣️: 10.

Here are the SQL queries for the given tasks:

(i) List the name and addresses of all employees who work for the IT department:

```sql
SELECT FirstName, LastName, address
FROM Employee
WHERE deptNo = (SELECT deptNo FROM Department WHERE deptName = 'IT');
```

(ii) List the total hours worked by each employee, arranged in order of department number and within department, alphabetically by employee surname:

```sql
SELECT e.FirstName, e.LastName, SUM(w.hours_worked) AS TotalHoursWorked
FROM Employee e
JOIN Work_on w ON e.empID = w.empID
GROUP BY e.FirstName, e.LastName
ORDER BY e.deptNo ASC, e.LastName ASC;
```

(iii) List the total number of employees in each department for those departments with more than 10 employees:

```sql
SELECT deptNo, COUNT(empID) AS EmployeeCount
FROM Employee
GROUP BY deptNo
HAVING COUNT(empID) > 10;
```

(iv) List the project number, project name, and the number of employees who work on that project:

```sql
SELECT projNo, projName, COUNT(empID) AS NumberOfEmployees
```

```
FROM Work_on
GROUP BY projNo, projName;
```

[20/03, 9:51 pm] *Meenakshi Arumugam*❣️: 11.

Here are the SQL queries for the given tasks:

i) SQL query to list all the machine allotments with the student names, lab, and machine numbers:

```sql
SELECT s.stud_name, l.description AS lab, a.mach_no
FROM Student s
JOIN Allotment a ON s.stud_no = a.stud_no
JOIN Lab l ON a.mach_no = l.mach_no;
```

ii) SQL query to list the total number of lab allotments daywise:

```sql
SELECT dayofweek, COUNT(*) AS total_lab_allotments
FROM Allotment
GROUP BY dayofweek;
```

iii) SQL query to give a count of how many machines have been allocated to the CSIT class:

```sql
SELECT COUNT(DISTINCT a.mach_no) AS machines_allocated
FROM Allotment a
JOIN Student s ON a.stud_no = s.stud_no
WHERE s.class = 'CSIT';
```

iv) SQL query to give machine allotment details of the stud_no 5 with his personal and class details:

```sql
SELECT s.stud_name, s.class, l.description AS lab, a.mach_no
FROM Student s
JOIN Allotment a ON s.stud_no = a.stud_no
JOIN Lab l ON a.mach_no = l.mach_no
WHERE s.stud_no = 5;
```

v) Create a view which lists the machine allotment details for Thursday:

```sql
CREATE VIEW ThursdayAllotments AS
SELECT s.stud_name, l.description AS lab, a.mach_no
FROM Student s
JOIN Allotment a ON s.stud_no = a.stud_no
JOIN Lab l ON a.mach_no = l.mach_no
WHERE a.dayofweek = 'Thursday';
```