Review 3

CURSOR (10):

1) **Retrieve employee details from the employee table** DELIMITER //

```
CREATE PROCEDURE GetEmployeeDetails()
BEGIN
  DECLARE emp_id INT;
  DECLARE emp_name VARCHAR(100);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Employee_ID, Employee_Name FROM employee;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;   OPEN cur;   read_loop: LOOP
    FETCH cur INTO emp_id, emp_name;
    IF done THEN LEAVE read_loop; END IF;
    SELECT emp_id, emp_name;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

emp_id	emp_name
3	Charlie Brown

2) **Get payroll details**

```
DELIMITER //
CREATE PROCEDURE GetPayrollDetails()
BEGIN
  DECLARE pay_id INT;
  DECLARE emp_id INT;
```

```
  DECLARE amount DECIMAL(10,2);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Payroll_ID, Employee_ID, Payroll_Final_Amount FROM payroll;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;   OPEN cur;   read_loop: LOOP
    FETCH cur INTO pay_id, emp_id, amount;
    IF done THEN LEAVE read_loop; END IF;
    SELECT pay_id, emp_id, amount;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

Result Grid | Filter Rows:

	pay_id	emp_id	amount
▶	2	NULL	38000.00

```
3)DELIMITER //
CREATE PROCEDURE GetBonuses()
BEGIN
  DECLARE bonus_id INT;
  DECLARE emp_id INT;
  DECLARE amount DECIMAL(10,2);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Bonus_ID, Employee_ID, Amount FROM bonus;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  OPEN cur;
read_loop: LOOP
    FETCH cur INTO bonus_id, emp_id, amount;
    IF done THEN LEAVE read_loop; END IF;
    SELECT bonus_id, emp_id, amount;
```

```
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

Result Grid | Filter Rows:

bonus_id	emp_id	amount
NULL	3	NULL

4)

```
DELIMITER //
CREATE PROCEDURE GetDeductions()
BEGIN
  DECLARE ded_id INT;
  DECLARE emp_id INT;
  DECLARE amount DECIMAL(10,2);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Deduction_ID, Employee_ID, Amount FROM deductions;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;   OPEN cur;   read_loop: LOOP
    FETCH cur INTO ded_id, emp_id, amount;
    IF done THEN LEAVE read_loop; END IF;
    SELECT ded_id, emp_id, amount;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

Result Grid | Filter Rows:

ded_id	emp_id	amount
3	NULL	NULL

5)DELIMITER //

CREATE PROCEDURE GetTaxes()

BEGIN

DECLARE tax_id INT;

DECLARE emp_id INT;

DECLARE amount DECIMAL(10,2);

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR SELECT Tax_ID, Employee_ID, Amount FROM tax;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =

TRUE; OPEN cur; read_loop: LOOP

FETCH cur INTO tax_id, emp_id, amount;

IF done THEN LEAVE read_loop; END IF;

SELECT tax_id, emp_id, amount;

END LOOP;

CLOSE cur;

END //

DELIMITER ;

6)DELIMITER //

CREATE PROCEDURE GetLeaves()

BEGIN

DECLARE leave_id INT;

DECLARE emp_id INT;

DECLARE leave_days INT;

```
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Leave_ID, Employee_ID, Total_Days FROM leave_details;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;   OPEN cur;   read_loop: LOOP
    FETCH cur INTO leave_id, emp_id, leave_days;
    IF done THEN LEAVE read_loop; END IF;
    SELECT leave_id, emp_id, leave_days;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

7)

```
DELIMITER //
CREATE PROCEDURE GetShiftDetails()
BEGIN
  DECLARE shift_id INT;
  DECLARE emp_id INT;
  DECLARE shift_name VARCHAR(50);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Shift_ID, Employee_ID, Shift_Name FROM shift;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;   OPEN cur;   read_loop: LOOP
    FETCH cur INTO shift_id, emp_id, shift_name;
    IF done THEN LEAVE read_loop; END IF;
    SELECT shift_id, emp_id, shift_name;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

Result Grid | Filter Rows: | Export:

shift_id	emp_id	shift_name
NULL	3	NULL

8)DELIMITER //

```
CREATE PROCEDURE GetEmployeeBankDetails()
BEGIN
  DECLARE bank_id INT;
  DECLARE emp_id INT;
  DECLARE acc_no VARCHAR(50);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Bank_ID, Employee_ID, Account_No FROM bank_details;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;   OPEN cur;   read_loop: LOOP
    FETCH cur INTO bank_id, emp_id, acc_no;
    IF done THEN LEAVE read_loop; END IF;
    SELECT bank_id, emp_id, acc_no;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

Result Grid | Filter Rows: | Export: | Wrap

bank_id	emp_id	acc_no
NULL	3	1122334455

9)DELIMITER //

```
CREATE PROCEDURE GetDepartmentDetails()
BEGIN
  DECLARE dept_id INT;
  DECLARE dept_name VARCHAR(100);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Department_ID, Department_Name FROM department;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;   OPEN cur;   read_loop: LOOP
    FETCH cur INTO dept_id, dept_name;
    IF done THEN LEAVE read_loop; END IF;
    SELECT dept_id, dept_name;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

10)DELIMITER //

```
CREATE PROCEDURE GetAttendanceRecords()
BEGIN
  DECLARE att_id INT;
  DECLARE emp_id INT;
  DECLARE check_in DATETIME;
  DECLARE check_out DATETIME;
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT Attendance_ID, Employee_ID, Check_in, Check_out FROM
attendance;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;   OPEN cur;   read_loop: LOOP
    FETCH cur INTO att_id, emp_id, check_in, check_out;
```

```
    IF done THEN LEAVE read_loop; END IF;
    SELECT att_id, emp_id, check_in, check_out;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```

Result Grid | Filter Rows:

	att_id	emp_id	check_in	check_out
▶	1	1	NULL	NULL

VIEW(10):

```
1)CREATE VIEW EmployeeSalary AS
SELECT e.Employee_ID, e.Employee_Name, s.Salary_Final_Amount
FROM employee e
JOIN salary s ON e.Employee_ID = s.Employee_ID;

2)CREATE VIEW PayrollDetails AS
SELECT e.Employee_Name, p.Payroll_Final_Amount, p.Payroll_Type
FROM employee e
JOIN payroll p ON e.Employee_ID = p.Employee_ID;

3)CREATE VIEW EmployeeAttendance AS
SELECT e.Employee_Name, a.Check_in, a.Check_out
FROM employee e
JOIN attendance a ON e.Employee_ID = a.Employee_ID;

4)CREATE VIEW EmployeeBonuses AS
SELECT e.Employee_Name, b.Amount AS BonusAmount
```

FROM employee e

JOIN bonus b ON e.Employee_ID = b.Employee_ID;

5)CREATE VIEW EmployeeDeductions AS

SELECT e.Employee_Name, d.Amount AS DeductionAmount

FROM employee e

JOIN deductions d ON e.Employee_ID = d.Employee_ID;

6)CREATE VIEW EmployeeTaxDetails AS

SELECT e.Employee_Name, t.Amount AS TaxAmount

FROM employee e

JOIN tax t ON e.Employee_ID = t.Employee_ID;

7)CREATE VIEW EmployeeShiftDetails AS

SELECT e.Employee_Name, s.Shift_Name

FROM employee e

JOIN shift s ON e.Employee_ID = s.Employee_ID;

8)CREATE VIEW DepartmentEmployees AS

SELECT d.Department_Name, e.Employee_Name

FROM department d

JOIN employee e ON d.Department_ID = e.Department_ID;

9)CREATE VIEW EmployeeBankDetails AS

SELECT e.Employee_Name, b.Account_No

FROM employee e

JOIN bank_details b ON e.Employee_ID = b.Employee_ID;

10)CREATE VIEW PayrollTransactions AS

SELECT p.Payroll_ID, e.Employee_Name, p.Payroll_Final_Amount, py.Payment_Mode

FROM payroll p

JOIN employee e ON p.Employee_ID = e.Employee_ID JOIN

payment py ON p.Payroll_ID = py.Payroll_ID;

Tables_in_payrolldb	Table_type
departmentemployees	VIEW
employeeattendance	VIEW
employeebankdetails	VIEW
employeebonuses	VIEW
employeedeductions	VIEW
employeedepartment	VIEW
employeepayrollview	VIEW
employeesalary	VIEW
payrolldetails	VIEW
payrolltransactions	VIEW

JOIN(10):

-- 1. Get employee names and their respective department names

SELECT e.Employee_Name, d.Department_Name

FROM employee e

JOIN department d ON e.Department_ID = d.Department_ID;

Result Grid | Filter Rows:

Employee_Name	Department_Name
Alice Johnson	HR
Bob Williams	IT
Ch Bob Williams	Finance

-- 2. Retrieve employee salaries along with their names

SELECT e.Employee_Name, s.Salary_Final_Amount

FROM employee e

JOIN salary s ON e.Employee_ID = s.Employee_ID;

Result Grid | Filter Rows:

Employee_Name	Salary_Final_Amount
Alice Johnson	50000.00
Bob Williams	60000.00
Charlie Brown	55000.00

-- 3. List employees with their respective bonuses (if any)

```
SELECT e.Employee_Name, COALESCE(b.Amount, 0) AS BonusAmount
FROM employee e
LEFT JOIN bonus b ON e.Employee_ID = b.Employee_ID;
```

Result Grid | Filter Rows:

Employee_Name	BonusAmount
Alice Johnson	2000.00
Bob Williams	2500.00
Charlie Brown	2200.00

-- 4. Show employees with any deductions

```
SELECT e.Employee_Name, COALESCE(d.Amount, 0) AS DeductionAmount
FROM employee e
LEFT JOIN deductions d ON e.Employee_ID = d.Employee_ID;
```

Result Grid | Filter Rows:

Employee_Name	DeductionAmount
Alice Johnson	0.00
Bob Williams	0.00
Charlie Brown	0.00

-- 5. Get employees with their assigned shifts

```
SELECT e.Employee_Name, s.Shift_Name
FROM employee e
JOIN shift s ON e.Employee_ID = s.Employee_ID;
```

Result Grid | Filter Rows:

Employee_Name	Shift_Type
Alice Johnson	Morning
Bob Williams	Evening
Charlie Brown	Night

-- 6. Retrieve employees with their tax deductions

SELECT e.Employee_Name, COALESCE(t.Amount, 0) AS TaxAmount

FROM employee e

LEFT JOIN tax t ON e.Employee_ID = t.Employee_ID;

Employee_Name	TaxAmount
Alice Johnson	0.00
Bob Williams	0.00
Charlie Brown	0.00

-- 7. Get all payroll details including employee names and payment modes SELECT p.Payroll_ID, e.Employee_Name, p.Payroll_Final_Amount, py.Payment_Mode

FROM payroll p

JOIN employee e ON p.Employee_ID = e.Employee_ID

JOIN payment py ON p.Payroll_ID = py.Payroll_ID;

Payroll_ID	Employee_Name	Payroll_Final_Amount	Payment_Mode

-- 8. Retrieve employees and their attendance records

SELECT e.Employee_Name, a.Check_in, a.Check_out

FROM employee e

JOIN attendance a ON e.Employee_ID = a.Employee_ID;

Result Grid | Filter Rows: | Export: | Wrap Cell

Employee_Name	Check_in	Check_out
Alice Johnson	2024-03-06 09:00:00	2024-03-06 18:00:00

-- 9. List all employees along with their bank details

```
SELECT e.Employee_Name, b.Account_No, b.Bank_Name
FROM employee e
JOIN bank_details b ON e.Employee_ID = b.Employee_ID;
```

Employee_Name	Account_No	Bank_ID
Alice Johnson	1234567890	1
Bob Williams	0987654321	2
Charlie Brown	1122334455	3

-- 10. Show all employees and their leave records

```
SELECT e.Employee_Name, l.Leave_Type, l.Total_Days
FROM employee e
JOIN leave_details l ON e.Employee_ID = l.Employee_ID;
```

Employee_Name	Leave_Type	Leave_desc
Alice Johnson	Sick Leave	Flu symptoms
Bob Williams	Casual Leave	Personal work
Charlie Brown	Annual Leave	Family vacation

GROUP BY(13):

-- 1. Count the number of employees in each department

```
SELECT Department_ID, COUNT(Employee_ID) AS EmployeeCount
FROM employee
GROUP BY Department_ID;
```

Result Grid | Filter Rows:

Department_ID	EmployeeCount
1	1
2	1
3	1

-- 2. Calculate the total payroll amount for each payroll type

```
SELECT Payroll_Type, SUM(Payroll_Final_Amount) AS TotalPayroll
FROM payroll
GROUP BY Payroll_Type;
```

Result Grid | Filter Rows:

Employee_ID	TotalPayroll
NULL	83000.00

-- 3. Find the average salary in each department

```
SELECT e.Department_ID, d.Department_Name, AVG(s.Salary_Final_Amount) AS AvgSalary
FROM employee e
JOIN salary s ON e.Employee_ID = s.Employee_ID
JOIN department d ON e.Department_ID = d.Department_ID GROUP BY
e.Department_ID, d.Department_Name;
```

Result Grid | Filter Rows: | Export

Department_ID	Department_Name	AvgSalary
1	HR	50000.000000
2	IT	60000.000000
3	Finance	55000.000000

-- 4. Count the number of employees per shift

```
SELECT s.Shift_Name, COUNT(e.Employee_ID) AS EmployeeCount
FROM employee e
```

```
JOIN shift s ON e.Employee_ID = s.Employee_ID
GROUP BY s.Shift_Name;
```

Shift_ID	EmployeeCount
1	1
2	1
3	1

```
-- 5. Calculate total bonus given per department
SELECT e.Department_ID, SUM(b.Amount) AS TotalBonus
FROM employee e
JOIN bonus b ON e.Employee_ID = b.Employee_ID
GROUP BY e.Department_ID;
```

Result Grid | Filter Rows:

Department_ID	TotalBonus
1	2000.00
2	2500.00
3	2200.00

```
-- 6. Find the total tax collected per department
SELECT e.Department_ID, SUM(t.Amount) AS TotalTax
FROM employee e
JOIN tax t ON e.Employee_ID = t.Employee_ID
GROUP BY e.Department_ID;
```

Department_ID	TotalTax

```
-- 7. Calculate the total deductions per employee
SELECT Employee_ID, SUM(Amount) AS TotalDeductions
FROM deductions
```

GROUP BY Employee_ID;

Employee_ID	TotalDeductions
NULL	4500.00

-- 8. Count the number of leave applications per leave type

SELECT Leave_Type, COUNT(Leave_ID) AS LeaveCount

FROM leave_details

GROUP BY Leave_Type;

Leave_Type	LeaveCount
Sick Leave	1
Casual Leave	1
Annual Leave	1

-- 9. Find the total payroll amount paid per employee

SELECT Employee_ID, SUM(Payroll_Final_Amount) AS TotalPayroll

FROM payroll

GROUP BY Employee_ID;

Result Grid | Filter Rows:

Employee_ID	TotalPayroll
NULL	83000.00

-- 10. Calculate the average check-in time per employee

SELECT Employee_ID, AVG(Check_in) AS AvgCheckInTime

FROM attendance

GROUP BY Employee_ID;

Result Grid | Filter Rows:

Employee_ID	AvgCheckInTime
1	20240306090000.0000

--11. SELECT Employee_ID, SUM(Amount) AS TotalBonus

FROM bonus

GROUP BY Employee_ID

HAVING TotalBonus > 1000;

Employees who received a bonus of more than $1000 in total

	Employee_ID	TotalBonus
▶	1	2000.00
	2	2500.00
	3	2200.00

--12. SELECT Employee_ID, COUNT(Bank_ID) AS AccountCount

FROM bank_details

GROUP BY Employee_ID

HAVING AccountCount > 0;

Employees with more than 1 bank accounts

	Employee_ID	AccountCount
▶	1	1
	2	1
	3	1

--13. SELECT Employee_ID, SUM(Payroll_Final_Amount) AS TotalPayrollPaid

FROM payroll

GROUP BY Employee_ID

HAVING TotalPayrollPaid > 7000;

Employees who received total payroll above $7000

	Employee_ID	TotalPayrollPaid
▶	NULL	83000.00

UNION(10):

-- 1. Get all employee IDs from both payroll and salary tables

SELECT Employee_ID FROM payroll

UNION

SELECT Employee_ID FROM salary;

Result Grid

Employee_ID
NULL
1
2
3

-- 2. List all employees who have received either a bonus or a deduction

```
SELECT Employee_ID, Amount AS Payment, 'Bonus' AS Type FROM bonus

UNION

SELECT Employee_ID, Amount AS Payment, 'Deduction' AS Type FROM deductions;
```

Employee_ID	Payment	Type
1	2000.00	Bonus
2	2500.00	Bonus
3	2200.00	Bonus
NULL	2000.00	Deduction
NULL	1500.00	Deduction
NULL	1000.00	Deduction

-- 3. Retrieve all unique employee names from payroll and bonus tables

```
SELECT e.Employee_Name FROM employee e

JOIN payroll p ON e.Employee_ID = p.Employee_ID

UNION

SELECT e.Employee_Name FROM employee e

JOIN bonus b ON e.Employee_ID = b.Employee_ID;
```

Employee_Name
Alice Johnson
Bob Williams
Charlie Brown

-- 4. Find all employees who have either tax deductions or bonuses

```
SELECT Employee_ID, Amount, 'Tax' AS Type FROM tax

UNION

SELECT Employee_ID, Amount, 'Bonus' AS Type FROM bonus;
```

Field	Type	Null
Bonus_ID	int	NO
Employee_ID	int	YES
Amount	decimal(10,2)	YES

-- 5. List all employees who are assigned a shift or have recorded attendance

```
SELECT Employee_ID FROM shift

UNION

SELECT Employee_ID FROM attendance;
```

Result Grid

Employee_ID
1
2
3

-- 6. Retrieve employee names from both salary and deductions tables

```
SELECT e.Employee_Name FROM employee e

JOIN salary s ON e.Employee_ID = s.Employee_ID

UNION

SELECT e.Employee_Name FROM employee e

JOIN deductions d ON e.Employee_ID = d.Employee_ID;
```

Result Grid Filter

Employee_Name
Alice Johnson
Bob Williams
Charlie Brown

-- 7. Find all employees who have either been assigned a shift or have a payroll record

```
SELECT Employee_ID FROM shift

UNION

SELECT Employee_ID FROM payroll;
```

Result Grid | Filter Ro

Employee_ID
1
2
3
NULL

-- 8. Get all department IDs from both employee and payroll tables

```
SELECT Department_ID FROM employee

UNION

SELECT Department_ID FROM payroll;
```

Field	Type
Department_ID	int
Department_Name	varchar(100)
Department_Location	varchar(255)

-- 9. List all employee names who have taken a leave or received a salary

```
SELECT e.Employee_Name FROM employee e

JOIN leave_details l ON e.Employee_ID = l.Employee_ID

UNION

SELECT e.Employee_Name FROM employee e

JOIN salary s ON e.Employee_ID = s.Employee_ID;
```

Result Grid | F

Employee_Name
Alice Johnson
Bob Williams
Charlie Brown

-- 10. Retrieve all employee IDs who have a bank account or received a payroll payment

```
SELECT Employee_ID FROM bank_details

UNION

SELECT Employee_ID FROM payroll;
```

Result Grid

Employee_ID
1
2
3
NULL

SUBQUERY(10):

-- 1. Get the employee with the highest salary

```
SELECT Employee_Name

FROM employee

WHERE Employee_ID = (SELECT Employee_ID FROM salary ORDER BY Salary_Final_Amount DESC
LIMIT 1);
```

Result Grid Filter Rows:

Employee_Name
Bob Williams

-- 2. Find employees who received a bonus greater than $1000

```
SELECT Employee_Name

FROM employee

WHERE Employee_ID IN (SELECT Employee_ID FROM bonus WHERE Amount > 1000);
```

Employee_Name
Alice Johnson
Bob Williams
Charlie Brown

-- 3. Retrieve employees who have deductions exceeding $500

```
SELECT Employee_Name
FROM employee
WHERE Employee_ID IN (SELECT Employee_ID FROM deductions WHERE Amount > 500);
```

	Employee_Name

-- 4. Get employees who have paid more than $2000 in taxes

```
SELECT Employee_Name
FROM employee
WHERE Employee_ID IN (SELECT Employee_ID FROM tax WHERE Amount > 2000);
```

	Employee_Name

-- 5. List employees who have not taken any leave

```
SELECT Employee_Name
FROM employee
WHERE Employee_ID NOT IN (SELECT DISTINCT Employee_ID FROM leave_details);
```

	Employee_Name
▶	Alice Johnson
	Bob Williams

-- 6. Get employees who have attended work at least once

```
SELECT Employee_Name
FROM employee
WHERE Employee_ID IN (SELECT DISTINCT Employee_ID FROM attendance);
```

Employee_Name
Alice Johnson

-- 7. Find employees who are not assigned any shifts

```
SELECT Employee_Name

FROM employee

WHERE Employee_ID NOT IN (SELECT DISTINCT Employee_ID FROM shift);
```

Result Grid

Employee_Name
Alice Johnson

-- 8. Get employees who received more in bonuses than deductions

```
SELECT Employee_Name  FROM

employee

WHERE Employee_ID IN (

   SELECT Employee_ID FROM bonus WHERE Amount >

   (SELECT COALESCE(SUM(Amount), 0) FROM deductions WHERE deductions.Employee_ID =
bonus.Employee_ID)

);
```

Employee_Name
Alice Johnson
Bob Williams
Charlie Brown

-- 9. Find departments that have more than 5 employees

```
SELECT Department_Name  FROM

department

WHERE Department_ID IN (
```

SELECT Department_ID FROM employee GROUP BY Department_ID HAVING COUNT(Employee_ID) > 0

);

Department_Name
HR
IT
Finance

-- 10. Get the payroll details of the employee who has the highest tax deduction

SELECT * FROM payroll

WHERE Employee_ID = (SELECT Employee_ID FROM tax ORDER BY Amount DESC LIMIT 1);

Employee_ID	TotalDeductions
NULL	4500.00

NESTED QUERIES(10):

-- 1. Get employees with a salary higher than the average salary

SELECT Employee_Name FROM employee

WHERE Employee_ID IN (SELECT Employee_ID FROM salary WHERE Salary_Final_Amount > (SELECT AVG(Salary_Final_Amount) FROM salary));

Employee_Name
Bob Williams

-- 2. Find employees who have received more bonuses than the average bonus

SELECT Employee_Name FROM employee

WHERE Employee_ID IN (SELECT Employee_ID FROM bonus WHERE Amount > (SELECT AVG(Amount) FROM bonus));

Employee_Name
Bob Williams

```
-- 3. Retrieve employees who have more deductions than the average deduction

SELECT Employee_Name FROM employee

WHERE Employee_ID IN (SELECT Employee_ID FROM deductions WHERE Amount > (SELECT
AVG(Amount) FROM deductions));


-- 4. Get employees with payroll amounts higher than the department's average payroll

SELECT Employee_Name FROM employee

WHERE Employee_ID IN (SELECT Employee_ID FROM payroll WHERE Payroll_Final_Amount
>  (SELECT AVG(Payroll_Final_Amount) FROM payroll));


-- 5. Find employees who have attended work more times than the average attendance

SELECT Employee_Name FROM employee

WHERE Employee_ID IN (SELECT Employee_ID FROM attendance GROUP BY Employee_ID
HAVING COUNT(*) >  (SELECT AVG(Check_in) FROM attendance));


-- 6. Retrieve employees whose tax deductions are in the top 10%

SELECT Employee_Name FROM employee

WHERE Employee_ID IN (SELECT Employee_ID FROM tax WHERE Amount > (SELECT
PERCENTILE_CONT(0.90) WITHIN GROUP (ORDER BY Amount) FROM tax));


-- 7. Find employees who have taken fewer leaves than the department average

SELECT Employee_Name FROM employee

WHERE Employee_ID IN (SELECT Employee_ID FROM leave_details GROUP BY Employee_ID HAVING
COUNT(*) <

   (SELECT AVG(Total_Days) FROM leave_details));


-- 8. Get the department with the highest average salary

SELECT Department_Name FROM department

WHERE Department_ID = (SELECT Department_ID FROM employee WHERE Employee_ID =

(SELECT Employee_ID FROM salary ORDER BY Salary_Final_Amount DESC LIMIT 1));
```

Department_Name
IT

-- 9. Find the shift that has the most employees assigned

```
SELECT Shift_Name FROM shift WHERE Shift_ID =
   (SELECT Shift_ID FROM employee GROUP BY Shift_ID ORDER BY COUNT(Employee_ID) DESC
LIMIT 1);
```

Shift_Type
Morning

-- 10. Get employees who work in the department with the most employees

```
SELECT Employee_Name FROM employee WHERE Department_ID =
   (SELECT Department_ID FROM employee GROUP BY Department_ID ORDER BY
COUNT(Employee_ID) DESC LIMIT 1);
```

Employee_Name
Alice Johnson

TRIGGER(10):

-- 1. Create a trigger to update payroll when salary changes

```
DELIMITER //
CREATE TRIGGER UpdatePayrollAfterSalaryChange
AFTER UPDATE ON salary
FOR EACH ROW
BEGIN
   UPDATE payroll
   SET Payroll_Final_Amount = NEW.Salary_Final_Amount
   WHERE Employee_ID = NEW.Employee_ID;
END;
//
```

```
DELIMITER ;

-- 2. Trigger to log employee deletions
DELIMITER //
CREATE TRIGGER LogEmployeeDeletion
BEFORE DELETE ON employee
FOR EACH ROW
BEGIN
   INSERT INTO employee_log (Employee_ID, Employee_Name, Action_Taken, Timestamp)
   VALUES (OLD.Employee_ID, OLD.Employee_Name, 'Deleted', NOW());
END;
//
DELIMITER ;
-- 3. Automatically insert a payroll entry when a new salary is added
DELIMITER //
CREATE TRIGGER InsertPayrollOnSalaryAddition
AFTER INSERT ON salary
FOR EACH ROW
BEGIN
   INSERT INTO payroll (Employee_ID, Payroll_Final_Amount, Payroll_Type)
   VALUES (NEW.Employee_ID, NEW.Salary_Final_Amount, 'Monthly');
END;
//
DELIMITER ;

-- 4. Prevent deleting departments that have employees
DELIMITER //
CREATE TRIGGER PreventDepartmentDeletion
BEFORE DELETE ON department
FOR EACH ROW
BEGIN
```

```
    IF (SELECT COUNT(*) FROM employee WHERE Department_ID = OLD.Department_ID) > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot delete department with employees.';
    END IF;
END;
//
DELIMITER ;
-- 5. Update attendance record when employee check-out is updated
DELIMITER //
CREATE TRIGGER UpdateAttendanceOnCheckOut
AFTER UPDATE ON attendance
FOR EACH ROW
BEGIN
    UPDATE attendance
    SET Check_out = NEW.Check_out
    WHERE Attendance_ID = NEW.Attendance_ID;
END;
//
DELIMITER ;

-- 6. Insert a bonus entry when an employee's salary crosses a threshold
DELIMITER //
CREATE TRIGGER AddBonusOnSalaryIncrease
AFTER UPDATE ON salary
FOR EACH ROW
BEGIN
    IF NEW.Salary_Final_Amount > 5000 THEN
        INSERT INTO bonus (Employee_ID, Amount) VALUES (NEW.Employee_ID, 500);
    END IF;
END;
//
DELIMITER ;
```

```
-- 7. Log tax deductions whenever they are updated
DELIMITER //
CREATE TRIGGER LogTaxChanges
AFTER UPDATE ON tax
FOR EACH ROW
BEGIN
   INSERT INTO tax_log (Employee_ID, Old_Amount, New_Amount, Change_Timestamp)     VALUES
(NEW.Employee_ID, OLD.Amount, NEW.Amount, NOW());
END;
//
DELIMITER ;

-- 8. Prevent deleting payroll records
DELIMITER //
CREATE TRIGGER PreventPayrollDeletion
BEFORE DELETE ON payroll
FOR EACH ROW
BEGIN
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Payroll records cannot be deleted.'; END;
//
DELIMITER ;
-- 9. Update leave record if an employee resigns
DELIMITER //
CREATE TRIGGER RemoveLeavesOnResignation
BEFORE DELETE ON employee
FOR EACH ROW
BEGIN
   DELETE FROM leave_details WHERE Employee_ID = OLD.Employee_ID;
END;
//
DELIMITER ;
```

-- 10. Automatically assign a default shift to new employees

```
DELIMITER //
CREATE TRIGGER AssignDefaultShift
AFTER INSERT ON employee
FOR EACH ROW
BEGIN
   INSERT INTO shift (Employee_ID, Shift_Name) VALUES (NEW.Employee_ID, 'Morning'); END;
//
DELIMITER ;
```

Trigger	Event	Table	Statement	Timing	Created	sql_mode
UpdateAttendanceOnCheckOut	UPDATE	attendance	BEGIN UPDATE attendance SET Check_ou...	AFTER	2025-04-02 16:12:10.36	ONLY_FULL_GROUP_BY,STRICT_TR
PreventDepartmentDeletion	DELETE	department	BEGIN IF (SELECT COUNT(*) FROM employe...	BEFORE	2025-04-02 16:12:06.64	ONLY_FULL_GROUP_BY,STRICT_TR
AssignDefaultShift	INSERT	employee	BEGIN INSERT INTO shift (Employee_ID, Shif...	AFTER	2025-04-02 16:12:32.80	ONLY_FULL_GROUP_BY,STRICT_TR
LogEmployeeDeletion	DELETE	employee	BEGIN INSERT INTO employee_log (Employe...	BEFORE	2025-04-02 16:10:16.07	ONLY_FULL_GROUP_BY,STRICT_TR
RemoveLeavesOnResignation	DELETE	employee	BEGIN DELETE FROM leave_details WHERE E...	BEFORE	2025-04-02 16:12:29.54	ONLY_FULL_GROUP_BY,STRICT_TR
PreventPayrollDeletion	DELETE	payroll	BEGIN SIGNAL SQLSTATE '45000' SET MESSA...	BEFORE	2025-04-02 16:12:25.35	ONLY_FULL_GROUP_BY,STRICT_TR
InsertPayrollOnSalaryAddition	INSERT	salary	BEGIN INSERT INTO payroll (Employee_ID, P...	AFTER	2025-04-02 12:52:38.97	ONLY_FULL_GROUP_BY,STRICT_TR
SalaryUpdate	UPDATE	salary	INSERT INTO salary_log (Salary_ID, Old_Amou...	AFTER	2025-04-01 00:00:26.86	ONLY_FULL_GROUP_BY,STRICT_TR
AddBonusOnSalaryIncrease	UPDATE	salary	BEGIN IF NEW.Salary_Final_Amount > 5000 ...	AFTER	2025-04-02 12:53:05.12	ONLY_FULL_GROUP_BY,STRICT_TR
UpdatePayrollAfterSalaryChange	UPDATE	salary	BEGIN UPDATE payroll SET Payroll_Final_...	AFTER	2025-04-02 16:09:45.40	ONLY_FULL_GROUP_BY,STRICT_TR

ORDER BY(10):

-- 1. Retrieve employees ordered by salary in descending order

```
SELECT Employee_Name, Salary_Final_Amount FROM employee
JOIN salary ON employee.Employee_ID = salary.Employee_ID
ORDER BY Salary_Final_Amount DESC;
```

Employee_Name	Salary_Final_Amount
Bob Williams	60000.00
Charlie Brown	55000.00
Alice Johnson	50000.00

-- 2. Get employees ordered by payroll amount (highest to lowest)

```
SELECT Employee_Name, Payroll_Final_Amount FROM employee
JOIN payroll ON employee.Employee_ID = payroll.Employee_ID
ORDER BY Payroll_Final_Amount DESC;
```

Employee_ID	Amount
2	2500.00
3	2200.00
1	2000.00

-- 3. List bonuses from highest to lowest

SELECT Employee_ID, Amount FROM bonus ORDER BY Amount DESC;

Employee_ID	Amount
2	2500.00
3	2200.00
1	2000.00

-- 4. Show deductions ordered by amount (highest first)

SELECT Employee_ID, Amount FROM deductions ORDER BY Amount DESC;

Result Grid | Filter Rows:

Employee_ID	Amount
NULL	2000.00
NULL	1500.00
NULL	1000.00

-- 5. Display tax records sorted by deduction amount

SELECT Employee_ID, Amount FROM tax ORDER BY Amount DESC;

Result Grid | Filter Rows:

Employee_ID	Tax_Amount
102	750.00
103	600.00
101	500.00

-- 6. Retrieve employees ordered by department name

SELECT Employee_Name, Department_ID FROM employee ORDER BY Department_ID ASC;

Employee_Name	Department_ID
Alice Johnson	1
Bob Williams	2
Charlie Brown	3

-- 7. Get attendance records sorted by check-in time

SELECT Employee_ID, Check_in FROM attendance ORDER BY Check_in ASC;

Employee_ID	Check_in
1	2024-03-06 09:00:00

-- 8. List employees sorted by bank account numbers

SELECT Employee_Name, Account_No FROM bank_details ORDER BY Account_No ASC;

Employee_ID	Account_No
2	0987654321
3	1122334455
1	1234567890

-- 9. Get employees with the highest payroll transactions first

SELECT Employee_Name, Payroll_Final_Amount FROM payroll ORDER BY Payroll_Final_Amount DESC;

Employee_ID	Payroll_Final_Amount
NULL	45000.00
NULL	38000.00

-- 10. Retrieve leave records sorted by LEAVE TYPE

SELECT Employee_ID, LEAVE_TYPE FROM leave_details ORDER BY LEAVE_TYPE DESC;

Employee_ID	leave_type
1	Sick Leave
2	Casual Leave
3	Annual Leave

HAVING CLAUSE(7):

-- 1. Get departments with more than 5 employees

SELECT Department_ID, COUNT(Employee_ID) AS EmployeeCount

FROM employee GROUP BY Department_ID HAVING EmployeeCount > 5;

Department_ID	EmployeeCount
1	1
2	1
3	1

-- 2. Find employees whose total payroll exceeds $5000

SELECT Employee_ID, SUM(Payroll_Final_Amount) AS TotalPayroll

FROM payroll GROUP BY Employee_ID HAVING TotalPayroll > 5000;

Employee_ID	TotalPayroll
NULL	83000.00

-- 3. List employees with average salary above $3000

SELECT Employee_ID, AVG(Salary_Final_Amount) AS AvgSalary

FROM salary GROUP BY Employee_ID HAVING AvgSalary > 3000;

Employee_ID	AvgSalary
1	50000.000000
2	60000.000000
3	55000.000000

-- 4. SELECT e.Department_ID, SUM(b.Amount) AS TotalBonus

FROM bonus b

JOIN employee e ON b.Employee_ID = e.Employee_ID

GROUP BY e.Department_ID

HAVING TotalBonus > 2000;

Result Grid | Filter Rows:

Department_ID	TotalBonus
2	2500.00
3	2200.00

-- 5. Employees with more than 3 leave requests

SELECT Employee_ID, COUNT(Leave_ID) AS LeaveCount

FROM leave_details GROUP BY Employee_ID HAVING LeaveCount > 3;

Employee_ID	LeaveCount
1	1
2	1
3	1

--6. Departments where the total salary expenditure exceeds $50,000 SELECT Salary_ID, SUM(Salary_Final_Amount) AS TotalSalary

FROM salary GROUP BY Salary_ID HAVING TotalSalary > 50000;

Salary_ID	TotalSalary
2	60000.00
3	55000.00

--7. Employees with more than 1 attendance records

SELECT Employee_ID, COUNT(Attendance_ID) AS AttendanceCount

FROM attendance GROUP BY Employee_ID HAVING AttendanceCount > 0;

	Employee_ID	AttendanceCount
▶	1	1