Mess Management System using SQL

```
DROP DATABASE IF EXISTS mess_management;
CREATE DATABASE mess_management;
USE mess_management;

CREATE TABLE members (
   member_id INT AUTO_INCREMENT PRIMARY KEY,
   full_name VARCHAR(100) NOT NULL,
   phone VARCHAR(20),
   role ENUM('member','manager') DEFAULT 'member',
   join_date DATE,
   status ENUM('active','inactive') DEFAULT 'active'
);

CREATE TABLE expenses (
   expense_id INT AUTO_INCREMENT PRIMARY KEY,
   member_id INT NULL,
   expense_date DATE NOT NULL,
   category VARCHAR(50),
   amount DECIMAL(10,2) NOT NULL,
   description VARCHAR(255),
   FOREIGN KEY (member_id) REFERENCES members(member_id)
      ON DELETE SET NULL
      ON UPDATE CASCADE
);

CREATE TABLE collections (
   collection_id INT AUTO_INCREMENT PRIMARY KEY,
   member_id INT NOT NULL,
   collection_date DATE NOT NULL,
   amount DECIMAL(10,2) NOT NULL,
   note VARCHAR(255),
   FOREIGN KEY (member_id) REFERENCES members(member_id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE payments (
   payment_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    member_id INT NULL,
    payment_date DATE NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    description VARCHAR(255),
    FOREIGN KEY (member_id) REFERENCES members(member_id)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

CREATE TABLE meals (
    meal_id INT AUTO_INCREMENT PRIMARY KEY,
    member_id INT NOT NULL,
    meal_date DATE NOT NULL,
    breakfast ENUM('yes','no') DEFAULT 'no',
    lunch ENUM('yes','no') DEFAULT 'no',
    dinner ENUM('yes','no') DEFAULT 'no',
    FOREIGN KEY (member_id) REFERENCES members(member_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE INDEX idx_members_role ON members(role);
CREATE INDEX idx_expense_member ON expenses(member_id);
CREATE INDEX idx_expense_date ON expenses(expense_date);
CREATE INDEX idx_expense_category ON expenses(category);
CREATE INDEX idx_collection_member ON collections(member_id);
CREATE INDEX idx_collection_date ON collections(collection_date);
CREATE INDEX idx_payment_date ON payments(payment_date);

INSERT INTO members (full_name, phone, role, join_date,status)
VALUES
('Rahim Uddin', '01700000001', 'manager', '2025-01-01', 'active'),
('Karim Ahmed', '01700000002', 'member', '2025-01-02', 'active'),
('Sumi Akter', '01700000003', 'member', '2025-01-05', 'active');
SELECT * FROM members;

INSERT INTO expenses (member_id, expense_date, category, amount, description)
VALUES
(1,'2025-01-15','Rent', 2000.00,'January rent'),
(2,'2025-01-15','Rent', 2000.00,'January rent'),
```

```
(3,'2025-01-15','Rent', 2000.00,'January rent'),

(1,'2025-01-10','Food', 2000.00,'Fish and Meat'),
(2,'2025-01-10','Food', 500.00,'Rice and vegetables'),
(3,'2025-01-10','Food', 400.00,'Fruits'),

(1,'2025-01-12','Utilities', 2000.00,'Electricity and Internet bill'),
(2,'2025-02-03','Food', 500.00,'February groceries');
SELECT * FROM expenses;

INSERT INTO collections (member_id, collection_date, amount, note)
VALUES
(1,'2025-01-7',5000.00,'Jan deposit'),
(2,'2025-01-7',5000.00,'Jan deposit'),
(3,'2025-01-7',5000.00,'Jan deposit'),
(2,'2025-02-7',5000.00,'Feb deposit');
SELECT * FROM collections;

INSERT INTO payments (member_id, payment_date, amount, description)
VALUES
(1,'2025-01-22', 6000.00,'Total Cost'),
(2,'2025-01-22',2500.00,'Total Cost'),
(3,'2025-01-22',2400.00,'Total Cost');
SELECT * FROM payments;

INSERT INTO meals (member_id, meal_date, breakfast, lunch, dinner)
VALUES
(1,'2025-01-15','yes','yes','no'),
(2,'2025-01-15','yes','no','yes'),
(3,'2025-01-15','no','yes','yes');

CREATE VIEW vw_member_monthly_expenses AS
SELECT member_id,
    DATE_FORMAT(expense_date,'%Y-%m') AS month,
    SUM(amount) AS total_expense
FROM expenses
GROUP BY member_id, month;

CREATE VIEW vw_member_monthly_payments AS
SELECT member_id,
```

```
    DATE_FORMAT(payment_date,'%Y-%m') AS month,
    SUM(amount) AS total_payment
FROM payments
GROUP BY member_id, month;

CREATE VIEW vw_member_meals AS
SELECT meal_id, member_id, meal_date, breakfast, lunch, dinner
FROM meals;

CREATE VIEW vw_member_ledger AS
SELECT m.member_id, m.full_name,
    IFNULL(SUM(e.amount),0) AS total_spent,
    IFNULL(SUM(c.amount),0) AS total_paid,
    IFNULL(SUM(c.amount),0) - IFNULL(SUM(e.amount),0) AS balance
FROM members m
LEFT JOIN expenses e ON m.member_id = e.member_id
LEFT JOIN collections c ON m.member_id = c.member_id
GROUP BY m.member_id;

CREATE TABLE monthly_summary_mv (
  month CHAR(7) PRIMARY KEY,
  total_expenses DECIMAL(12,2) DEFAULT 0,
  total_collections DECIMAL(12,2) DEFAULT 0
);

INSERT INTO monthly_summary_mv (month, total_expenses, total_collections)
SELECT month,
    SUM(total_expenses) AS total_expenses,
    SUM(total_collections) AS total_collections
FROM (
  SELECT DATE_FORMAT(expense_date,'%Y-%m') AS month,
      SUM(amount) AS total_expenses,
      0 AS total_collections
  FROM expenses
  GROUP BY month
  UNION ALL
  SELECT DATE_FORMAT(collection_date,'%Y-%m') AS month,
      0 AS total_expenses,
      SUM(amount) AS total_collections
  FROM collections
```

```
   GROUP BY month
) summary
GROUP BY month;

SELECT m.full_name, e.category, e.amount
FROM members m INNER JOIN expenses e
ON m.member_id = e.member_id;

SELECT m.full_name, c.amount
FROM members m LEFT JOIN collections c
ON m.member_id = c.member_id;

SELECT m.full_name, p.amount
FROM members m RIGHT JOIN payments p
ON m.member_id = p.member_id;

SELECT * FROM members NATURAL JOIN collections;

SELECT member_id, SUM(amount) total
FROM expenses
GROUP BY member_id
HAVING SUM(amount) > 1500;

SELECT * FROM expenses ORDER BY amount DESC;

SELECT * FROM expenses
WHERE expense_date BETWEEN '2025-01-01' AND '2025-01-31'
ORDER BY expense_id;
SELECT * FROM payments
WHERE payment_date BETWEEN '2025-01-01' AND '2025-01-31'
ORDER BY payment_id;

SELECT member_id, full_name, total_spent, total_paid,
      CASE
        WHEN total_paid - total_spent > 0 THEN 'Paid'
        WHEN total_paid - total_spent < 0 THEN 'Due'
        ELSE 'settled'
      END AS status
FROM vw_member_ledger
ORDER BY (total_paid - total_spent) ASC;
```

```
SELECT * FROM members WHERE role = 'manager';

SELECT IF(COUNT(*) = 1, 'ALLOW', 'BLOCKED') AS permission
FROM members
WHERE member_id = 1 AND role = 'manager';

SELECT IF(COUNT(*) = 0, 'ALLOW', 'BLOCKED') AS permission
FROM members
WHERE member_id = 2 AND role = 'member';
```