Solution of 1

```
Drop database if exists lab_final;
create database lab_final;
use lab_final;

CREATE TABLE Company (
    company_id INT PRIMARY KEY,
    company_name VARCHAR(100) NOT NULL,
    jurisdiction VARCHAR(100),
    risk_score INT,
    balance_millions DECIMAL(10,2)
);

-- Table 2: Transfers (Transaction)
CREATE TABLE Transaction (
    transfer_id INT PRIMARY KEY AUTO_INCREMENT,

    amount DECIMAL(18, 2) NOT NULL,
    transaction_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    from_company_id INT NOT NULL,
    FOREIGN KEY (from_company_id) REFERENCES Company(company_id),
    to_company_id INT NOT NULL,
    FOREIGN KEY (to_company_id) REFERENCES Company(company_id),
    CONSTRAINT chk_different_companies
        CHECK (from_company_id <> to_company_id),
    CONSTRAINT chk_positive_amount
        CHECK (amount > 0)
);

INSERT INTO Company (company_id, company_name, jurisdiction, risk_score,
balance_millions) VALUES
(101, 'Alpha Holdings', 'Cayman Is.', 65, 50.00),
(102, 'Beta LLC', 'Switzerland', 20, 12.50),
(103, 'Gamma Corp', 'Cayman Is.', 90, 5.00),
(104, 'Delta Inc', 'Panama', 99, 6.99),
(105, 'Epsilon Ltd', 'Switzerland', 15, 100.00),
(106, 'Zeta Global', 'Singapore', 10, 75.20),
(107, 'Eta Ventures', 'UK', 10, 2.00),
(108, 'Theta Shell', 'Panama', 99, 10.00);

INSERT INTO Transaction (transfer_id, from_company_id, to_company_id, amount,
transaction_date) VALUES
(1, 101, 102, 10.00, '2023-01-01'),
(2, 102, 103, 8.00, '2023-01-02'),
(3, 103, 101, 7.50, '2023-01-03'),
(4, 104, 105, 20.00, '2023-01-04'),
(5, 105, 104, 19.50, '2023-01-05'),
(6, 101, 106, 50.00, '2023-01-06'),
(7, 106, 107, 45.00, '2023-01-07'),
(8, 107, 108, 40.00, '2023-01-08'),
(9, 108, 101, 38.00, '2023-01-09'),
(10, 102, 105, 2.00, '2023-01-10'),
(11, 103, 107, 1.00, '2023-01-11'),
(12, 106, 102, 5.00, '2023-01-12');
```

Solution of 2

```
-- Solve of 2
use lab_final;
WITH CompanyTotalVolume AS (
   -- Step 1: Calculate the total amount transacted (sent + received) for each company
   SELECT
      c.company_id,
      c.company_name,
      -- Using SUM() directly, assuming transactions always exist, or that NULL sum for a
company means 0.
      -- We'll keep COALESCE(SUM(t.amount), 0) for robustness, but simplify the join
      SUM(CASE
            WHEN t.from_company_id = c.company_id OR t.to_company_id = c.company_id
            THEN t.amount
            ELSE 0
         END) AS total_transaction_amount
   FROM
      Company c
   -- Standard JOIN might miss companies with no transactions entirely,
   -- so LEFT JOIN to Transaction is safer for calculating total volume for ALL companies.
   LEFT JOIN
      Transaction t ON c.company_id = t.from_company_id OR c.company_id = t.to_company_id
   GROUP BY
      c.company_id, c.company_name
),
RankedVolume AS (
   -- Step 2: Rank the companies based on their total transaction amount
   SELECT
      company_id,
        company_name,
        total_transaction_amount,
        -- Using RANK() as requested, which may create gaps in the ranking.
        RANK() OVER (ORDER BY total_transaction_amount DESC) AS rank_num
    FROM
        CompanyTotalVolume
)
-- Step 3: Select the companies at the 2nd and 4th rank
SELECT
    company_id,
    company_name,
    total_transaction_amount
FROM
    RankedVolume
WHERE
    rank_num IN (2, 4);
```

Solution of 3

```
use lab_final;
WITH RECURSIVE TransferPaths AS (
    SELECT
        from_company_id AS source_company_id,
        to_company_id AS current_company_id,
        CONVERT(from_company_id, CHAR(255)) AS path_trace,
        1 AS step_count
    FROM
        Transaction
    UNION ALL
    SELECT
        tp.source_company_id,
        t.to_company_id AS current_company_id,
        CONCAT(tp.path_trace, '->', t.to_company_id) AS path_trace,
        tp.step_count + 1
    FROM
        TransferPaths tp
    JOIN
        Transaction t ON tp.current_company_id = t.from_company_id
    WHERE
        LOCATE(t.to_company_id, tp.path_trace) = 0
        AND tp.step_count < 10
)
SELECT
```

```
    tp.source_company_id,
    t.to_company_id AS loop_end_company_id,
    CONCAT(tp.path_trace, '->', t.to_company_id) AS looping_path
FROM
    TransferPaths tp
JOIN
    Transaction t ON tp.current_company_id = t.from_company_id
WHERE
    t.to_company_id = tp.source_company_id
ORDER BY
    tp.source_company_id, looping_path;
```