Complex SQL Query Explanations

1. Total Number of Appointments by Month

Query:

```
SELECT
    DATE_FORMAT(appointment_date, '%Y-%m') AS month,
    COUNT(*) AS total_appointments
FROM
    Appointments
WHERE
    status != 'Cancelled'
GROUP BY
    DATE_FORMAT(appointment_date, '%Y-%m')
ORDER BY
    month DESC;
```

Explanation:

- Aggregates the number of appointments per month.
- Filters out cancelled appointments.
- Useful for time-based trend analysis.

Key Points:

- Uses DATE_FORMAT() for monthly grouping.
- Helps hospital management analyze trends.
- Simple and efficient aggregation.

2. Past Appointments with Pending Payments

Query:

```
SELECT
    p.patient_id, CONCAT(p.first_name, ' ', p.last_name) AS patient_name,
    a.appointment_id, at.type_name AS appointment_type, a.appointment_date,
    CONCAT(d.first_name, ' ', d.last_name) AS doctor_name, dep.department_name,
    py.amount AS total_charge, py.amount_paid, py.insurance_covered,
    (py.amount - py.amount_paid - py.insurance_covered) AS balance_due,
    py.status AS payment_status,
    DATEDIFF(CURDATE(), a.appointment_date) AS days_overdue
FROM
    Payments py
    JOIN Appointments a ON py.appointment_id = a.appointment_id
    JOIN Patients p ON a.patient_id = p.patient_id
    JOIN Doctors d ON a.doctor_id = d.doctor_id
```

```
    JOIN Departments dep ON d.department_id = dep.department_id
    JOIN AppointmentTypes at ON a.type_id = at.type_id
WHERE
    py.status = 'Pending'
    AND a.appointment_date < CURDATE()
    AND (py.amount - py.amount_paid - py.insurance_covered) > 0
ORDER BY
    a.appointment_date ASC;
```

Explanation:
- Finds overdue payments for completed appointments.
- Calculates exact balance still due.
- Uses DATEDIFF to show how long overdue.

Key Points:
- Financial tracking and recovery.
- Multi-table joins with filters.
- Practical in billing and collections.

3. Total Revenue by Doctor

Query:

```
SELECT
    d.doctor_id, CONCAT(d.first_name, ' ', d.last_name) AS doctor_name,
    d.specialization, COUNT(DISTINCT a.appointment_id) AS total_appointments,
    (SUM(py.amount_paid) + SUM(py.insurance_covered)) AS total_revenue
FROM
    Doctors d
    JOIN Departments dep ON d.department_id = dep.department_id
    LEFT JOIN Appointments a ON d.doctor_id = a.doctor_id
    LEFT JOIN Payments py ON a.appointment_id = py.appointment_id
WHERE
    a.status != 'Cancelled'
    AND py.status IN ('Completed', 'Partial')
GROUP BY
    d.doctor_id, doctor_name, d.specialization, dep.department_name
ORDER BY
    total_revenue DESC;
```

Explanation:
- Displays doctor-wise revenue from appointments.

- Includes both paid and insurance amounts.
- Handles doctors with 0 revenue using LEFT JOIN.

Key Points:
- Ideal for doctor performance tracking.
- Includes safeguards with filters.
- Summarizes financial impact clearly.

4. Patients Who Spent the Most

Query:

```
SELECT
    P.patient_id, P.first_name, P.last_name, SUM(Pay.amount) AS total_spent
FROM
    Patients P
JOIN
    Appointments A ON P.patient_id = A.patient_id
JOIN
    Payments Pay ON A.appointment_id = Pay.appointment_id
GROUP BY
    P.patient_id
ORDER BY
    total_spent DESC
LIMIT 5;
```

Explanation:
- Sums total spend per patient across all appointments.
- Filters top 5 spenders using LIMIT.

Key Points:
- Highlights high-value patients.
- Useful for personalized services or rewards.
- Straightforward and efficient query.