5 Triggers

1. Trigger: Log Patient Insertions

```
CREATE TABLE patient_log (
   log_id SERIAL PRIMARY KEY,
   patient_id INT,
   action TEXT,
   log_time TIMESTAMP
);

-- Step 2: Function
CREATE OR REPLACE FUNCTION log_patient_insert()
RETURNS TRIGGER AS $$
BEGIN
   INSERT INTO patient_log (patient_id, action, log_time)
   VALUES (NEW.patient_id, 'INSERT', CURRENT_TIMESTAMP);
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Step 3: Trigger
CREATE TRIGGER trg_log_patient_insert
AFTER INSERT ON PATIENT
FOR EACH ROW
EXECUTE FUNCTION log_patient_insert();
```

2. Trigger: Auto-set Admission Date

```
CREATE OR REPLACE FUNCTION set_admission_date()
RETURNS TRIGGER AS $$
BEGIN
   NEW.admission_date := CURRENT_DATE;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_auto_admission_date
BEFORE INSERT ON INPATIENT
FOR EACH ROW
EXECUTE FUNCTION set_admission_date();
```

3. Trigger: Prevent Double Discharge

```
CREATE OR REPLACE FUNCTION prevent_double_discharge()
RETURNS TRIGGER AS $$
BEGIN
  IF OLD.discharge_date IS NOT NULL THEN
    RAISE EXCEPTION 'Patient already discharged';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_prevent_double_discharge
BEFORE UPDATE OF discharge_date ON INPATIENT
FOR EACH ROW
EXECUTE FUNCTION prevent_double_discharge();
```

4. Trigger: Update Billing on Test Insert

```
CREATE OR REPLACE FUNCTION update_bill_after_test()
RETURNS TRIGGER AS $$
BEGIN
  UPDATE BILL
  SET test_charge = COALESCE(test_charge, 0) + NEW.cost,
    total_amount = COALESCE(total_amount, 0) + NEW.cost
  WHERE patient_id = NEW.patient_id;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_update_bill_after_test
AFTER INSERT ON MEDICAL_TEST
```

```
FOR EACH ROW
EXECUTE FUNCTION update_bill_after_test();
```

5. Trigger: Log Doctor Assignment

```
CREATE TABLE doctor_assignment_log (
   log_id SERIAL PRIMARY KEY,
   patient_id INT,
   doctor_id INT,
   assignment_time TIMESTAMP
);

CREATE OR REPLACE FUNCTION log_doctor_assignment()
RETURNS TRIGGER AS $$
BEGIN
   INSERT INTO doctor_assignment_log (patient_id, doctor_id,
assignment_time)
   VALUES (NEW.patient_id, NEW.doctor_id, CURRENT_TIMESTAMP);
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_log_doctor_assignment
AFTER INSERT ON PATIENT_DOCTOR
FOR EACH ROW
EXECUTE FUNCTION log_doctor_assignment();
```