Hospital Patient Records Cleaning and Automation

1. Database and Schema Creation

SQL:

```
CREATE DATABASE IF NOT EXISTS HOSPITAL_DB;
USE DATABASE HOSPITAL_DB;
CREATE SCHEMA IF NOT EXISTS PATIENT_SCHEMA;
```

Explanation:
- HOSPITAL_DB stores all hospital data.
- PATIENT_SCHEMA organizes related tables.
- Keeping separate schemas for raw and cleaned data improves auditability, consistency, and debugging.

2. Source and Target Table Design

SQL:

```
CREATE OR REPLACE TABLE RAW_PATIENTS (
  patient_id VARCHAR,
  name VARCHAR,
  age VARCHAR,
  gender VARCHAR,
  visit_date VARCHAR,
  diagnosis_code VARCHAR,
  bill_amount VARCHAR,
  inserted_at TIMESTAMP_LTZ DEFAULT CURRENT_TIMESTAMP()
);

CREATE OR REPLACE TABLE CLEAN_PATIENTS (
  patient_id INTEGER,
  name VARCHAR,
  age INTEGER,
  gender VARCHAR,
  visit_date DATE,
  diagnosis_code VARCHAR,
  bill_amount FLOAT,
  processed_at TIMESTAMP_LTZ DEFAULT CURRENT_TIMESTAMP()
);
```

Explanation:
- RAW_PATIENTS accepts any data type safely.
- CLEAN_PATIENTS enforces correct types for analytics.
- processed_at shows when cleaning occurred, enabling auditing and freshness tracking.

3. Stream Creation

SQL:

```
CREATE OR REPLACE STREAM RAW_PATIENTS_STREAM
ON TABLE RAW_PATIENTS
APPEND_ONLY = FALSE;
```

Explanation:
A stream records all data changes in RAW_PATIENTS.
It enables incremental, automated cleaning by feeding new rows to the cleaning task, ensuring real-time updates.

4. Data Cleaning Task

SQL (Task Body):

```
INSERT INTO CLEAN_PATIENTS (patient_id, name, age, gender, visit_date, diagnosis_code, bill_amount, processed_at)
SELECT
 TRY_TO_NUMBER(r.patient_id)::INT,
 r.name,
 COALESCE(
  TRY_TO_NUMBER(REGEXP_REPLACE(COALESCE(r.age,''), '[^0-9]', ''))::INT,
  CASE
   WHEN LOWER(r.age) LIKE '%twenty-five%' OR LOWER(r.age) LIKE '%twenty five%' THEN 25
   WHEN LOWER(r.age) LIKE '%twenty%' THEN 20
   WHEN LOWER(r.age) LIKE '%thirty%' THEN 30
   WHEN LOWER(r.age) LIKE '%forty%' THEN 40
   WHEN LOWER(r.age) LIKE '%fifty%' THEN 50
   WHEN LOWER(r.age) LIKE '%sixty%' THEN 60
   ELSE 0
  END
 ) AS age,
 r.gender,
 COALESCE(
  TRY_TO_DATE(r.visit_date, 'YYYY-MM-DD'),
  TRY_TO_DATE(r.visit_date, 'DD-MM-YYYY'),
  TRY_TO_DATE(r.visit_date, 'MM/DD/YYYY'),
  TRY_TO_DATE(r.visit_date, 'YYYY/MM/DD'),
  CURRENT_DATE()
 ) AS visit_date,
 r.diagnosis_code,
 COALESCE(
  TRY_TO_NUMBER(REGEXP_REPLACE(COALESCE(r.bill_amount,''), '[^0-9.]', ''))::FLOAT,
  0.0
```

```
 ) AS bill_amount,
 CURRENT_TIMESTAMP()
FROM RAW_PATIENTS_STREAM s
JOIN RAW_PATIENTS r ON s.PATIENT_ID = r.PATIENT_ID
WHERE s.METADATA$ACTION = 'INSERT' AND s.METADATA$ISUPDATE = FALSE;
```

Explanation:
- TRY_TO_NUMBER safely converts text numbers.
- TRY_TO_DATE handles multiple formats.
- COALESCE and CASE ensure fallbacks for missing values.
- Ensures consistent, clean, typed data for reporting.

5. Task Scheduling

SQL:

```
CREATE OR REPLACE TASK PATIENT_CLEANING_TASK
 WAREHOUSE = MY_TASK_WAREHOUSE
 SCHEDULE = 'USING CRON */10 * * * * UTC'
AS
-- include cleaning SQL here
;

ALTER TASK PATIENT_CLEANING_TASK RESUME;
```

Explanation:
Scheduling every 10 minutes automates cleaning, reduces manual work, and ensures fresh, reliable data for analysis.

6. Inserting Messy Data

SQL:

```
INSERT INTO RAW_PATIENTS (patient_id, name, age, gender, visit_date, diagnosis_code,
bill_amount)
VALUES
('1', 'John Doe', '30', 'Male', '2025-10-21', 'D01', '5,000'),
('2', 'Jane Smith', 'Twenty-Five', 'Female', '21-10-2025', 'D02', '3,200'),
('3', 'Alex Brown', NULL, 'Male', '2025/10/22', 'D03', 'abc'),
('4', 'Mary Lee', '40', NULL, NULL, 'D04', NULL);
```

Explanation:
Issues fixed by cleaning task:
- Text ages ('Twenty-Five') → numeric.
- Bill with commas → numeric 5000.
- Invalid bill ('abc') → 0.
- Mixed date formats → standardized.
- Missing fields handled gracefully with defaults.

7. Manual Task Execution

SQL:

```
EXECUTE TASK PATIENT_CLEANING_TASK;
```

Explanation:
Manual execution is used for testing, backfilling, or running after maintenance when the scheduled task is paused.

8. Verification

SQL:

```
SELECT * FROM CLEAN_PATIENTS ORDER BY patient_id;
```

Checks:
- Age and bill_amount are numeric.
- Dates parsed correctly.
- processed_at is current.
- Row count matches new raw inserts.

9. Extension - Missing Diagnosis Flagging

SQL:

```
CREATE OR REPLACE TABLE PATIENTS_MISSING_DIAGNOSIS AS
SELECT * FROM CLEAN_PATIENTS WHERE 1=0;

INSERT INTO PATIENTS_MISSING_DIAGNOSIS
SELECT * FROM CLEAN_PATIENTS
WHERE diagnosis_code IS NULL OR TRIM(diagnosis_code) = '';
```

Explanation:
Adds a review table to flag missing diagnosis_code records for staff to correct manually.

10. Reflection

Explanation:
Snowflake stream-task pipelines process data incrementally as soon as changes occur.
Streams track changes; tasks automate transformations.
Compared to batch ETL, it offers:
- Real-time updates
- Less infrastructure
- Automatic failure recovery

This ensures accurate, up-to-date hospital data for analytics and reporting.