Script A — Create and extend ENUM types (run alone, no BEGIN/COMMIT)

```
-- ========================================
-- Script A: Ensure ENUM types exist
-- Run this entire script alone (no BEGIN/COMMIT)
-- ========================================

-- 1) subject_type
DO $$
BEGIN
   IF NOT EXISTS (SELECT 1 FROM pg_type WHERE typname = 'subject_type') THEN
      CREATE TYPE subject_type AS ENUM ('Compulsory', 'Elective',
'International-Custom');
   END IF;
END
$$ LANGUAGE plpgsql;

-- 2) assessment_type
DO $$
BEGIN
   IF NOT EXISTS (SELECT 1 FROM pg_type WHERE typname = 'assessment_type')
THEN
      CREATE TYPE assessment_type AS ENUM ('Formative', 'Summative', 'Mixed');
   END IF;
END
$$ LANGUAGE plpgsql;

-- 3) curriculum_type base (create if missing)
DO $$
BEGIN
   IF NOT EXISTS (SELECT 1 FROM pg_type WHERE typname = 'curriculum_type') THEN
      CREATE TYPE curriculum_type AS ENUM ('Nursery', 'Primary-Local',
'Secondary-LSC', 'International');
   END IF;
END
$$ LANGUAGE plpgsql;

-- 4) curriculum_type extensions (add new values if missing)
DO $$
BEGIN
   IF NOT EXISTS (
      SELECT 1 FROM pg_enum e JOIN pg_type t ON e.enumtypid = t.oid
      WHERE t.typname = 'curriculum_type' AND e.enumlabel = 'Nursery-Primary-Local'
   ) THEN
      ALTER TYPE curriculum_type ADD VALUE 'Nursery-Primary-Local';
   END IF;
```

```
    IF NOT EXISTS (
        SELECT 1 FROM pg_enum e JOIN pg_type t ON e.enumtypid = t.oid
        WHERE t.typname = 'curriculum_type' AND e.enumlabel = 'International-Primary'
    ) THEN
        ALTER TYPE curriculum_type ADD VALUE 'International-Primary';
    END IF;

    IF NOT EXISTS (
        SELECT 1 FROM pg_enum e JOIN pg_type t ON e.enumtypid = t.oid
        WHERE t.typname = 'curriculum_type' AND e.enumlabel = 'International-Secondary'
    ) THEN
        ALTER TYPE curriculum_type ADD VALUE 'International-Secondary';
    END IF;
END
$$ LANGUAGE plpgsql;
```

Tip: If any part throws an error, execute `ROLLBACK;` to clear the aborted transaction state, fix the issue, then rerun Script A.

Script B — Full SAR module schema (BEGIN/COMMIT)

```
-- ============================================================
-- Script B: Bigezo SAR Module Migration (Version 1.4)
-- Purpose: Implement exams, marks, and reporting module
-- Target: PostgreSQL (PgAdmin)
-- Prerequisite: Script A ran successfully
-- ============================================================

BEGIN;

-- STEP 1: Modify existing students table (if present)
DO $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM information_schema.tables
        WHERE table_schema = 'public' AND table_name = 'students'
    ) THEN
        BEGIN
            EXECUTE 'ALTER TABLE students ADD COLUMN IF NOT EXISTS lin_number
VARCHAR(255) UNIQUE';
        EXCEPTION WHEN duplicate_column THEN
            NULL;
        END;
    END IF;
```

```
END
$$ LANGUAGE plpgsql;

-- STEP 2: Master NCDC reference table
CREATE TABLE IF NOT EXISTS ref_ncdc_lsc_subjects (
    ncdc_ref_id SERIAL PRIMARY KEY,
    subject_name VARCHAR(255) NOT NULL UNIQUE,
    s1_s2_mandatory BOOLEAN NOT NULL DEFAULT FALSE,
    s3_s4_mandatory BOOLEAN NOT NULL DEFAULT FALSE
);

-- STEP 3: Configuration tables

CREATE TABLE IF NOT EXISTS config_grading_scales (
    scale_id SERIAL PRIMARY KEY,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE,
    grade_letter VARCHAR(10) NOT NULL,
    descriptor TEXT,
    min_score_percent NUMERIC(5,2) NOT NULL,
    CONSTRAINT unique_grade_def_per_school UNIQUE (school_id, grade_letter)
);

CREATE TABLE IF NOT EXISTS config_school_settings (
    setting_id SERIAL PRIMARY KEY,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE
UNIQUE,
    curriculum_type curriculum_type NOT NULL,
    grading_scale_ref INT REFERENCES config_grading_scales(scale_id) ON DELETE SET
NULL,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS config_subjects (
    subject_id SERIAL PRIMARY KEY,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE,
    subject_name VARCHAR(255) NOT NULL,
    school_level VARCHAR(50) NOT NULL, -- 'Baby','Middle','Top','P.1'...'P.7','S.1'...'S.6','Year
1'...'Year 13 (A-Level/IB 2)'
    subject_type subject_type NOT NULL,
    ncdc_reference_name VARCHAR(255),
    max_selections_allowed INT DEFAULT 1,
    CONSTRAINT unique_school_subject_level UNIQUE (school_id, subject_name,
school_level)
);

CREATE TABLE IF NOT EXISTS config_exam_sets (
    exam_set_id SERIAL PRIMARY KEY,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE,
```

```
    set_name VARCHAR(100) NOT NULL, -- e.g., 'Mid Term Exams'
    class_level VARCHAR(50) NOT NULL, -- e.g., 'S.3', 'P.7', 'Year 10 (IGCSE)', 'Baby'
    term INT NOT NULL,
    year INT NOT NULL,
    assessment_type assessment_type NOT NULL,
    CONSTRAINT unique_exam_set_context UNIQUE (school_id, set_name, class_level,
term, year)
);

CREATE TABLE IF NOT EXISTS config_assessment_elements (
    element_id SERIAL PRIMARY KEY,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE,
    subject_id INT NOT NULL REFERENCES config_subjects(subject_id) ON DELETE
CASCADE,
    exam_set_id INT NOT NULL REFERENCES config_exam_sets(exam_set_id) ON
DELETE CASCADE,
    element_name VARCHAR(100) NOT NULL, -- e.g., 'Paper 1', 'Project'
    max_score INT NOT NULL,
    contributing_weight_percent NUMERIC(5,2) NOT NULL
);

CREATE TABLE IF NOT EXISTS config_holistic_metrics (
    metric_id SERIAL PRIMARY KEY,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE,
    metric_type VARCHAR(50), -- 'Value', 'Generic Skill', 'Life Skill', 'Developmental Domain'
    metric_name VARCHAR(255) NOT NULL,
    CONSTRAINT unique_metric_name_per_school UNIQUE (school_id, metric_name)
);

-- STEP 4: Transactional and results tables

CREATE TABLE IF NOT EXISTS results_header (
    header_id SERIAL PRIMARY KEY,
    student_id INT NOT NULL REFERENCES students(student_id) ON DELETE CASCADE,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE,
    subject_id INT NOT NULL REFERENCES config_subjects(subject_id) ON DELETE
CASCADE,
    term INT NOT NULL,
    year INT NOT NULL,
    CONSTRAINT unique_result_per_student_subject UNIQUE (student_id, subject_id, term,
year)
);

CREATE TABLE IF NOT EXISTS results_exam_entries (
    exam_entry_id SERIAL PRIMARY KEY,
    student_id INT NOT NULL REFERENCES students(student_id) ON DELETE CASCADE,
    subject_id INT NOT NULL REFERENCES config_subjects(subject_id) ON DELETE
CASCADE,
```

```
    exam_set_id INT NOT NULL REFERENCES config_exam_sets(exam_set_id) ON
DELETE CASCADE,
    status VARCHAR(50) DEFAULT 'Pending Entry',
    CONSTRAINT unique_student_subject_exam_set UNIQUE (student_id, subject_id,
exam_set_id)
);

CREATE TABLE IF NOT EXISTS results_entry (
    entry_id SERIAL PRIMARY KEY,
    exam_entry_id INT NOT NULL REFERENCES results_exam_entries(exam_entry_id) ON
DELETE CASCADE,
    element_id INT NOT NULL REFERENCES config_assessment_elements(element_id) ON
DELETE CASCADE,
    score_obtained NUMERIC(5,2) NOT NULL,
    max_score_at_entry INT NOT NULL,
    entered_by_user_id INT REFERENCES users(user_id) ON DELETE SET NULL,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    CONSTRAINT unique_element_score UNIQUE (exam_entry_id, element_id)
);

CREATE TABLE IF NOT EXISTS reports_summary (
    summary_id SERIAL PRIMARY KEY,
    header_id INT NOT NULL REFERENCES results_header(header_id) ON DELETE
CASCADE UNIQUE,
    total_percentage_score NUMERIC(5,2) NOT NULL,
    final_grade_ref INT REFERENCES config_grading_scales(scale_id),
    weighted_formative_score NUMERIC(5,2),
    weighted_summative_score NUMERIC(5,2),
    class_teacher_comment TEXT,
    head_teacher_comment TEXT
);

CREATE TABLE IF NOT EXISTS reports_holistic_feedback (
    feedback_id SERIAL PRIMARY KEY,
    student_id INT NOT NULL REFERENCES students(student_id) ON DELETE CASCADE,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE,
    term INT NOT NULL,
    year INT NOT NULL,
    metric_id INT NOT NULL REFERENCES config_holistic_metrics(metric_id) ON DELETE
CASCADE,
    rating VARCHAR(50),
    CONSTRAINT unique_holistic_rating UNIQUE (student_id, metric_id, term, year)
);

CREATE TABLE IF NOT EXISTS report_documents (
    doc_id SERIAL PRIMARY KEY,
    student_id INT NOT NULL REFERENCES students(student_id) ON DELETE CASCADE,
    school_id INT NOT NULL REFERENCES schools(school_id) ON DELETE CASCADE,
```

```
    term INT NOT NULL,
    year INT NOT NULL,
    file_path TEXT NOT NULL,
    generated_at TIMESTAMPTZ DEFAULT NOW(),
    CONSTRAINT unique_report_document UNIQUE (student_id, term, year)
);

-- STEP 5: Indexes

CREATE INDEX IF NOT EXISTS idx_config_subjects_school_level
    ON config_subjects (school_id, school_level);

CREATE INDEX IF NOT EXISTS idx_results_exam_entries_student_set
    ON results_exam_entries (student_id, exam_set_id);

CREATE INDEX IF NOT EXISTS idx_reports_summary_header
    ON reports_summary (header_id);

CREATE INDEX IF NOT EXISTS idx_holistic_feedback_student
    ON reports_holistic_feedback (student_id);

COMMIT;
```

Verify after running

- Check the enums:

```
SELECT e.enumlabel
FROM pg_enum e JOIN pg_type t ON e.enumtypid = t.oid
WHERE t.typname = 'curriculum_type'
ORDER BY e.enumsortorder;
```

- Check key tables exist:

```
SELECT table_name FROM information_schema.tables
WHERE table_schema = 'public'
AND table_name IN (
  'config_school_settings','config_subjects','config_exam_sets',
  'config_assessment_elements','config_grading_scales','config_holistic_metrics',
  'results_header','results_exam_entries','results_entry',
  'reports_summary','reports_holistic_feedback','report_documents'
)
ORDER BY table_name;
```