DATABASE MANAGEMENT SYSTEM

TELEVISION MANAGEMENT SYSTEM

TEAM MEMBERS:

KRISHNA SHREE B

NILA PR

SAMYUKTHA K

ER DIAGRAM:



APP:

# app.py

from flask import Flask, render\_template, request, redirect, url\_for, flash

import psycopg2

import psycopg2.extras

app = Flask(\_\_name\_\_)

app.secret\_key = "change-me-to-a-secret"

def get\_connection():

    return psycopg2.connect(

        host="localhost",

        database="television\_management",  # change if needed

        user="postgres",

        password="Nila@125",

    )

# ---------- HOME ----------

@app.route('/')

def home():

    return render\_template('home.html')

# ---------- CRUD helpers for each table ----------

# For brevity I will implement full CRUD for Channel, Genre, Show, CastAndCrew, Marketing, Advertisement, Episode, UpcomingShow.

# Each route follows a pattern: list, add (POST), update (POST), delete (GET)

# CHANNEL

@app.route('/channels')

def list\_channels():

    conn = get\_connection()

    cur = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

    cur.execute("SELECT \* FROM Channel ORDER BY channel\_id;")

    rows = cur.fetchall()

    cur.close(); conn.close()

    return render\_template('channels.html', channels=rows)

@app.route('/channels/add', methods=['POST'])

def add\_channel():

    name = request.form['channel\_name']

    satellite = request.form.get('satellite') or None

    num\_of\_shows = request.form.get('num\_of\_shows') or None

    location = request.form.get('location') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("INSERT INTO Channel(channel\_name, satellite, num\_of\_shows, location) VALUES (%s,%s,%s,%s)",

                (name, satellite, num\_of\_shows, location))

    conn.commit(); cur.close(); conn.close()

    flash('Channel added')

    return redirect(url\_for('list\_channels'))

@app.route('/channels/update/<int:id>', methods=['POST'])

def update\_channel(id):

    name = request.form['channel\_name']

    satellite = request.form.get('satellite') or None

    num\_of\_shows = request.form.get('num\_of\_shows') or None

    location = request.form.get('location') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("UPDATE Channel SET channel\_name=%s, satellite=%s, num\_of\_shows=%s, location=%s WHERE channel\_id=%s",

                (name, satellite, num\_of\_shows, location, id))

    conn.commit(); cur.close(); conn.close()

    flash('Channel updated')

    return redirect(url\_for('list\_channels'))

@app.route('/channels/delete/<int:id>')

def delete\_channel(id):

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("DELETE FROM Channel WHERE channel\_id=%s",(id,))

    conn.commit(); cur.close(); conn.close()

    flash('Channel deleted')

    return redirect(url\_for('list\_channels'))

# GENRE

@app.route('/genres')

def list\_genres():

    conn = get\_connection(); cur = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

    cur.execute("SELECT \* FROM Genre ORDER BY genre\_id;")

    rows = cur.fetchall(); cur.close(); conn.close()

    return render\_template('genre.html', genres=rows)

@app.route('/genres/add', methods=['POST'])

def add\_genre():

    name = request.form['genre\_name']

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("INSERT INTO Genre(genre\_name) VALUES (%s)", (name,))

    conn.commit(); cur.close(); conn.close()

    flash('Genre added'); return redirect(url\_for('list\_genres'))

@app.route('/genres/update/<int:id>', methods=['POST'])

def update\_genre(id):

    name = request.form['genre\_name']

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("UPDATE Genre SET genre\_name=%s WHERE genre\_id=%s", (name,id))

    conn.commit(); cur.close(); conn.close()

    flash('Genre updated'); return redirect(url\_for('list\_genres'))

@app.route('/genres/delete/<int:id>')

def delete\_genre(id):

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("DELETE FROM Genre WHERE genre\_id=%s",(id,))

    conn.commit(); cur.close(); conn.close()

    flash('Genre deleted'); return redirect(url\_for('list\_genres'))

# SHOW

@app.route('/shows')

def list\_shows():

    conn = get\_connection(); cur = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

    cur.execute("""SELECT s.\*, c.channel\_name, g.genre\_name

                   FROM Show s

                   LEFT JOIN Channel c ON s.channel\_id = c.channel\_id

                   LEFT JOIN Genre g ON s.genre\_id = g.genre\_id

                   ORDER BY s.show\_id;""")

    rows = cur.fetchall(); cur.close(); conn.close()

    return render\_template('show.html', shows=rows)

@app.route('/shows/add', methods=['POST'])

def add\_show():

    channel\_id = request.form.get('channel\_id') or None

    genre\_id = request.form.get('genre\_id') or None

    title = request.form['title']

    air\_time = request.form.get('air\_time') or None

    director\_id = request.form['director\_id']

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("""INSERT INTO Show(channel\_id, genre\_id, title, air\_time, director\_id)

                   VALUES (%s,%s,%s,%s,%s)""",

                (channel\_id, genre\_id, title, air\_time, director\_id))

    conn.commit(); cur.close(); conn.close()

    flash('Show added'); return redirect(url\_for('list\_shows'))

@app.route('/shows/update/<int:id>', methods=['POST'])

def update\_show(id):

    channel\_id = request.form.get('channel\_id') or None

    genre\_id = request.form.get('genre\_id') or None

    title = request.form['title']

    air\_time = request.form.get('air\_time') or None

    director\_id = request.form.get('director\_id') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("""UPDATE Show SET channel\_id=%s, genre\_id=%s, title=%s, air\_time=%s, director\_id=%s WHERE show\_id=%s""",

                (channel\_id, genre\_id, title, air\_time, director\_id, id))

    conn.commit(); cur.close(); conn.close()

    flash('Show updated'); return redirect(url\_for('list\_shows'))

@app.route('/shows/delete/<int:id>')

def delete\_show(id):

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("DELETE FROM Show WHERE show\_id=%s",(id,))

    conn.commit(); cur.close(); conn.close()

    flash('Show deleted'); return redirect(url\_for('list\_shows'))

# CAST AND CREW

@app.route('/castcrew')

def list\_castcrew():

    conn = get\_connection(); cur = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

    cur.execute("SELECT \* FROM CastAndCrew ORDER BY castcrew\_id;")

    rows = cur.fetchall(); cur.close(); conn.close()

    return render\_template('castcrew.html', castcrew=rows)

@app.route('/castcrew/add', methods=['POST'])

def add\_castcrew():

    name = request.form['name']; role\_type = request.form['role\_type']; show\_id = request.form.get('show\_id') or None; role\_description = request.form.get('role\_description') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("INSERT INTO CastAndCrew(name, role\_type, show\_id, role\_description) VALUES (%s,%s,%s,%s)",

                (name, role\_type, show\_id, role\_description))

    conn.commit(); cur.close(); conn.close()

    flash('Person added'); return redirect(url\_for('list\_castcrew'))

@app.route('/castcrew/update/<int:id>', methods=['POST'])

def update\_castcrew(id):

    name = request.form['name']; role\_type = request.form['role\_type']; show\_id = request.form.get('show\_id') or None; role\_description = request.form.get('role\_description') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("UPDATE CastAndCrew SET name=%s, role\_type=%s, show\_id=%s, role\_description=%s WHERE castcrew\_id=%s",

                (name, role\_type, show\_id, role\_description, id))

    conn.commit(); cur.close(); conn.close()

    flash('Person updated'); return redirect(url\_for('list\_castcrew'))

@app.route('/castcrew/delete/<int:id>')

def delete\_castcrew(id):

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("DELETE FROM CastAndCrew WHERE castcrew\_id=%s",(id,))

    conn.commit(); cur.close(); conn.close()

    flash('Person deleted'); return redirect(url\_for('list\_castcrew'))

# MARKETING

@app.route('/marketing')

def list\_marketing():

    conn = get\_connection(); cur = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

    cur.execute("""SELECT m.\*, s.title, c.channel\_name FROM Marketing m

                   LEFT JOIN Show s ON m.show\_id = s.show\_id

                   LEFT JOIN Channel c ON m.channel\_id = c.channel\_id

                   ORDER BY m.marketing\_id;""")

    rows = cur.fetchall(); cur.close(); conn.close()

    return render\_template('marketing.html', marketing=rows)

@app.route('/marketing/add', methods=['POST'])

def add\_marketing():

    channel\_id = request.form.get('channel\_id') or None

    show\_id = request.form.get('show\_id') or None

    trp = request.form.get('trp') or None

    cost = request.form.get('cost\_of\_production') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("INSERT INTO Marketing(channel\_id, show\_id, trp, cost\_of\_production) VALUES (%s,%s,%s,%s)",

                (channel\_id, show\_id, trp, cost))

    conn.commit(); cur.close(); conn.close()

    flash('Marketing added'); return redirect(url\_for('list\_marketing'))

@app.route('/marketing/delete/<int:id>')

def delete\_marketing(id):

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("DELETE FROM Marketing WHERE marketing\_id=%s",(id,))

    conn.commit(); cur.close(); conn.close()

    flash('Marketing deleted'); return redirect(url\_for('list\_marketing'))

# ADVERTISEMENT

@app.route('/ads')

def list\_ads():

    conn = get\_connection(); cur = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

    cur.execute("SELECT a.\*, c.channel\_name FROM Advertisement a LEFT JOIN Channel c ON a.channel\_id=c.channel\_id ORDER BY a.ad\_id;")

    rows = cur.fetchall(); cur.close(); conn.close()

    return render\_template('ads.html', ads=rows)

@app.route('/ads/add', methods=['POST'])

def add\_ad():

    channel\_id = request.form.get('channel\_id') or None

    company = request.form.get('company\_name')

    num\_of\_ads = request.form.get('num\_of\_ads') or None

    duration = request.form.get('duration') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("INSERT INTO Advertisement(channel\_id, company\_name, num\_of\_ads, duration) VALUES (%s,%s,%s,%s)",

                (channel\_id, company, num\_of\_ads, duration))

    conn.commit(); cur.close(); conn.close()

    flash('Ad added'); return redirect(url\_for('list\_ads'))

@app.route('/ads/delete/<int:id>')

def delete\_ad(id):

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("DELETE FROM Advertisement WHERE ad\_id=%s",(id,))

    conn.commit(); cur.close(); conn.close()

    flash('Ad deleted'); return redirect(url\_for('list\_ads'))

# EPISODE

@app.route('/episodes')

def list\_episodes():

    conn = get\_connection(); cur = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

    cur.execute("SELECT e.\*, s.title FROM Episode e LEFT JOIN Show s ON e.show\_id = s.show\_id ORDER BY e.episode\_id;")

    rows = cur.fetchall(); cur.close(); conn.close()

    return render\_template('episodes.html', episodes=rows)

@app.route('/episodes/add', methods=['POST'])

def add\_episode():

    show\_id = request.form.get('show\_id') or None

    season = request.form.get('season\_number') or None

    episode\_number = request.form.get('episode\_number') or None

    duration = request.form.get('duration') or None

    air\_date = request.form.get('air\_date') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("INSERT INTO Episode(show\_id, season\_number, episode\_number, duration, air\_date) VALUES (%s,%s,%s,%s,%s)",

                (show\_id, season, episode\_number, duration, air\_date))

    conn.commit(); cur.close(); conn.close()

    flash('Episode added'); return redirect(url\_for('list\_episodes'))

@app.route('/episodes/delete/<int:id>')

def delete\_episode(id):

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("DELETE FROM Episode WHERE episode\_id=%s",(id,))

    conn.commit(); cur.close(); conn.close()

    flash('Episode deleted'); return redirect(url\_for('list\_episodes'))

# UPCOMING SHOWS

@app.route('/upcoming')

def list\_upcoming():

    conn = get\_connection(); cur = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

    cur.execute("SELECT u.\*, c.channel\_name FROM UpcomingShow u LEFT JOIN Channel c ON u.channel\_id = c.channel\_id ORDER BY u.upcoming\_id;")

    rows = cur.fetchall(); cur.close(); conn.close()

    return render\_template('upcoming.html', upcoming=rows)

@app.route('/upcoming/add', methods=['POST'])

def add\_upcoming():

    channel\_id = request.form.get('channel\_id') or None

    title = request.form.get('title')

    replacement = request.form.get('replacement\_show\_id') or None

    director\_id = request.form.get('director\_id') or None

    conn = get\_connection(); cur = conn.cursor()

    cur.execute("INSERT INTO UpcomingShow(channel\_id, title, replacement\_show\_id, director\_id) VALUES (%s,%s,%s,%s)",

                (channel\_id, title, replacement, director\_id))

    conn.commit(); cur.close(); conn.close()

    flash('Upcoming show added'); return redirect(url\_for('list\_upcoming'))

@app.route('/upcoming/promote/<int:id>')

def promote\_upcoming(id):

    conn = get\_connection(); cur = conn.cursor()

    try:

        cur.execute("CALL promote\_upcoming\_show(%s);", (id,))

        conn.commit()

        flash('Upcoming show promoted to Show')

    except Exception as e:

        flash(f'Error: {e}')

    finally:

        cur.close(); conn.close()

    return redirect(url\_for('list\_upcoming'))

# ---------- Run canned complex queries (the 20) ----------

@app.route('/queries')

def queries\_page():

    return render\_template('queries.html')

@app.route('/run\_named\_query', methods=['POST'])

def run\_named\_query():

    name = request.form['query\_name']

    conn = get\_connection(); cur = conn.cursor()

    try:

        if name == 'avg\_trp\_per\_genre':

            cur.execute("""SELECT g.genre\_name, ROUND(AVG(m.trp)::numeric,2) AS avg\_trp

                           FROM Genre g

                           JOIN Show s ON s.genre\_id = g.genre\_id

                           JOIN Marketing m ON m.show\_id = s.show\_id

                           GROUP BY g.genre\_name;""")

        elif name == 'top3\_expensive\_per\_channel':

            cur.execute("""SELECT channel\_id, title, cost\_of\_production FROM (

                              SELECT c.channel\_id, s.title, m.cost\_of\_production,

                                     ROW\_NUMBER() OVER (PARTITION BY c.channel\_id ORDER BY m.cost\_of\_production DESC) rn

                              FROM Channel c

                              JOIN Show s ON s.channel\_id = c.channel\_id

                              JOIN Marketing m ON m.show\_id = s.show\_id

                            ) x WHERE rn <= 3;""")

        elif name == 'ad\_duration\_per\_channel':

            cur.execute("""SELECT c.channel\_name, COALESCE(SUM(a.duration), '00:00:00') AS total\_ad\_duration

                           FROM Channel c

                           LEFT JOIN Advertisement a ON a.channel\_id = c.channel\_id

                           GROUP BY c.channel\_name;""")

        elif name == 'shows\_trp\_above9\_cost\_below\_avg':

            cur.execute("""WITH avg\_cost AS (SELECT AVG(cost\_of\_production) AS avgc FROM Marketing)

                           SELECT s.title, m.trp, m.cost\_of\_production

                           FROM Show s

                           JOIN Marketing m ON m.show\_id = s.show\_id

                           WHERE m.trp > 9 AND m.cost\_of\_production < (SELECT avgc FROM avg\_cost);""")

        elif name == 'directors\_no\_show\_below5':

            cur.execute("""SELECT d.name, d.castcrew\_id

                           FROM CastAndCrew d

                           WHERE d.role\_type = 'Director'

                             AND EXISTS (SELECT 1 FROM Show s WHERE s.director\_id = d.castcrew\_id)

                             AND NOT EXISTS (

                               SELECT 1 FROM Show s

                               JOIN Marketing m ON s.show\_id = m.show\_id

                               WHERE s.director\_id = d.castcrew\_id AND m.trp < 5

                             );""")

        elif name == 'channel\_biggest\_cost\_diff':

            cur.execute("""SELECT channel\_id, channel\_name, (max\_cost - min\_cost) AS diff

                           FROM (

                             SELECT c.channel\_id, c.channel\_name,

                                    MAX(m.cost\_of\_production) AS max\_cost,

                                    MIN(m.cost\_of\_production) AS min\_cost

                             FROM Channel c

                             JOIN Show s ON s.channel\_id = c.channel\_id

                             JOIN Marketing m ON m.show\_id = s.show\_id

                             GROUP BY c.channel\_id, c.channel\_name

                           ) t

                           ORDER BY diff DESC

                           LIMIT 1;""")

        elif name == 'channel\_summary\_view':

            cur.execute("SELECT \* FROM channel\_summary;")

        elif name == 'generate\_weekly\_schedule':

            # call procedure as example for channel 1 and date today

            cur.execute("CALL generate\_weekly\_schedule(%s, %s);", (1, '2025-10-27'))

            conn.commit()

            cur.execute("SELECT \* FROM ShowSchedule WHERE channel\_id = 1 ORDER BY scheduled\_date, scheduled\_time;")

        elif name == 'shows\_short\_high\_trp':

            cur.execute("""SELECT sd.title, sd.total\_duration, m.trp

                           FROM show\_details sd

                           JOIN Marketing m ON sd.show\_id = m.show\_id

                           WHERE sd.total\_duration < INTERVAL '00:30:00' AND m.trp > 8;""")

        elif name == 'trp\_above\_avg':

            cur.execute("""WITH avg\_trp AS (SELECT AVG(trp) as avg\_trp FROM Marketing)

                           SELECT s.title, m.trp FROM Show s JOIN Marketing m ON s.show\_id = m.show\_id

                           WHERE m.trp > (SELECT avg\_trp FROM avg\_trp);""")

        elif name == 'cost\_per\_genre\_channel':

            cur.execute("""SELECT c.channel\_name, g.genre\_name, SUM(m.cost\_of\_production) AS total\_cost

                           FROM Marketing m

                           JOIN Show s ON m.show\_id = s.show\_id

                           JOIN Channel c ON c.channel\_id = s.channel\_id

                           JOIN Genr g ON g.genre\_id = s.genre\_id

                           GROUP BY c.channel\_name, g.genre\_name

                           ORDER BY c.channel\_name, total\_cost DESC;""")

        elif name == 'directors\_multiple\_genres':

            cur.execute("""SELECT d.castcrew\_id, d.name, COUNT(DISTINCT s.genre\_id) AS num\_genres

                           FROM CastAndCrew d

                           JOIN Show s ON s.director\_id = d.castcrew\_id

                           GROUP BY d.castcrew\_id, d.name

                           HAVING COUNT(DISTINCT s.genre\_id) > 1;""")

        elif name == 'directors\_also\_actors\_shows':

            cur.execute("""WITH duals AS (

                             SELECT name, castcrew\_id FROM CastAndCrew

                             WHERE name IN (SELECT name FROM CastAndCrew WHERE role\_type = 'Actor')

                               AND name IN (SELECT name FROM CastAndCrew WHERE role\_type = 'Director')

                           )

                           SELECT s.title, d.name AS director

                           FROM Show s

                           JOIN CastAndCrew d ON s.director\_id = d.castcrew\_id

                           WHERE d.castcrew\_id IN (SELECT castcrew\_id FROM duals);""")

        elif name == 'avg\_episode\_duration\_per\_season':

            cur.execute("""SELECT show\_id, season\_number, AVG(duration) as avg\_duration

                           FROM Episode

                           GROUP BY show\_id, season\_number

                           ORDER BY show\_id, season\_number;""")

        elif name == 'upcoming\_replacing\_low\_trp':

            cur.execute("""SELECT u.upcoming\_id, u.title, s.title AS replaced\_title, m.trp

                           FROM UpcomingShow u

                           LEFT JOIN Show s ON u.replacement\_show\_id = s.show\_id

                           LEFT JOIN Marketing m ON s.show\_id = m.show\_id

                           WHERE m.trp < 5 OR m.trp IS NULL;""")

        elif name == 'prevent\_trp\_trigger\_test':

            # Attempt to insert a bad TRP to show trigger works (will raise). Insert handled in app form normally.

            cur.execute("INSERT INTO Marketing(channel\_id, show\_id, trp, cost\_of\_production) VALUES (1,1,11,1000);")

            conn.commit()

            cur.execute("SELECT 'should not reach here';")

        elif name == 'add\_show\_with\_validation':

            # example call

            cur.execute("CALL add\_show\_valid(%s,%s,%s,%s,%s);", (1, None, 'New show from app', None, None))

            conn.commit()

            cur.execute("SELECT 'Added (check shows)';")

        else:

            cur.execute("SELECT 'Unknown query';")

        rows = cur.fetchall()

    except Exception as e:

        conn.rollback()

        rows = [(f'Error: {e}',)]

    finally:

        cur.close(); conn.close()

    return render\_template('query\_result.html', rows=rows, name=name)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

DATABASE:

import psycopg2

def get\_connection():

    return psycopg2.connect(

        host="localhost",

        database="television\_management",

        user="postgres",

        password="\*\*\*\*\*\*\*\*"

    )

HOME:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>TV Management Dashboard</title>

  <style>

    body {

      margin: 0;

      padding: 0;

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      height: 100vh;

      background: url('mis.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

      color: white;

      overflow: hidden;

    }

    .overlay {

      background: rgba(0, 0, 0, 0.6);

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

      z-index: 0;

    }

    .container {

      position: relative;

      z-index: 1;

      text-align: center;

      padding: 40px;

      border-radius: 20px;

      backdrop-filter: blur(6px);

      background: rgba(255, 255, 255, 0.1);

      box-shadow: 0 0 40px rgba(0, 0, 0, 0.3);

      width: 90%;

      max-width: 1000px;

    }

    h1 {

      font-size: 36px;

      margin-bottom: 30px;

      color: white;

      text-shadow: 2px 2px 8px rgba(0, 0, 0, 0.6);

      letter-spacing: 1px;

      animation: fadeIn 1s ease-in-out;

    }

    .menu {

      display: grid;

      grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));

      gap: 20px;

      justify-items: center;

      padding: 20px;

    }

    .card {

      background: rgba(255, 255, 255, 0.15);

      border: 1px solid rgba(255, 255, 255, 0.3);

      border-radius: 16px;

      padding: 20px;

      width: 200px;

      text-align: center;

      transition: all 0.3s ease;

      cursor: pointer;

      color: white;

      font-weight: 600;

      box-shadow: 0 0 15px rgba(255, 255, 255, 0.1);

    }

    .card:hover {

      transform: translateY(-8px);

      background: rgba(255, 255, 255, 0.25);

      box-shadow: 0 0 20px rgba(255, 255, 255, 0.4);

    }

    a {

      text-decoration: none;

      color: inherit;

    }

    .footer {

      margin-top: 30px;

      font-size: 14px;

      opacity: 0.8;

    }

    /\* Fade-in animation for smooth entry \*/

    @keyframes fadeIn {

      from { opacity: 0; transform: translateY(-10px); }

      to { opacity: 1; transform: translateY(0); }

    }

    /\* Flash messages styling \*/

    .flash {

      background: rgba(255, 255, 255, 0.2);

      color: white;

      padding: 10px;

      margin-top: 15px;

      border-radius: 8px;

      list-style: none;

    }

    .flash li {

      margin: 5px 0;

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h1>📺 TV Management Dashboard</h1>

    <div class="menu">

      <a href="/channels"><div class="card">📡 Channels</div></a>

      <a href="/genres"><div class="card">🎭 Genres</div></a>

      <a href="/shows"><div class="card">🎬 Shows</div></a>

      <a href="/castcrew"><div class="card">👥 Cast & Crew</div></a>

      <a href="/marketing"><div class="card">💼 Marketing</div></a>

      <a href="/ads"><div class="card">📢 Advertisements</div></a>

      <a href="/episodes"><div class="card">🎞️ Episodes</div></a>

      <a href="/upcoming"><div class="card">⏳ Upcoming Shows</div></a>

      <a href="/queries"><div class="card">🧠 Complex Queries</div></a>

    </div>

    {% with messages = get\_flashed\_messages() %}

      {% if messages %}

        <ul class="flash">

          {% for m in messages %}

            <li>{{ m }}</li>

          {% endfor %}

        </ul>

      {% endif %}

    {% endwith %}

    <div class="footer">

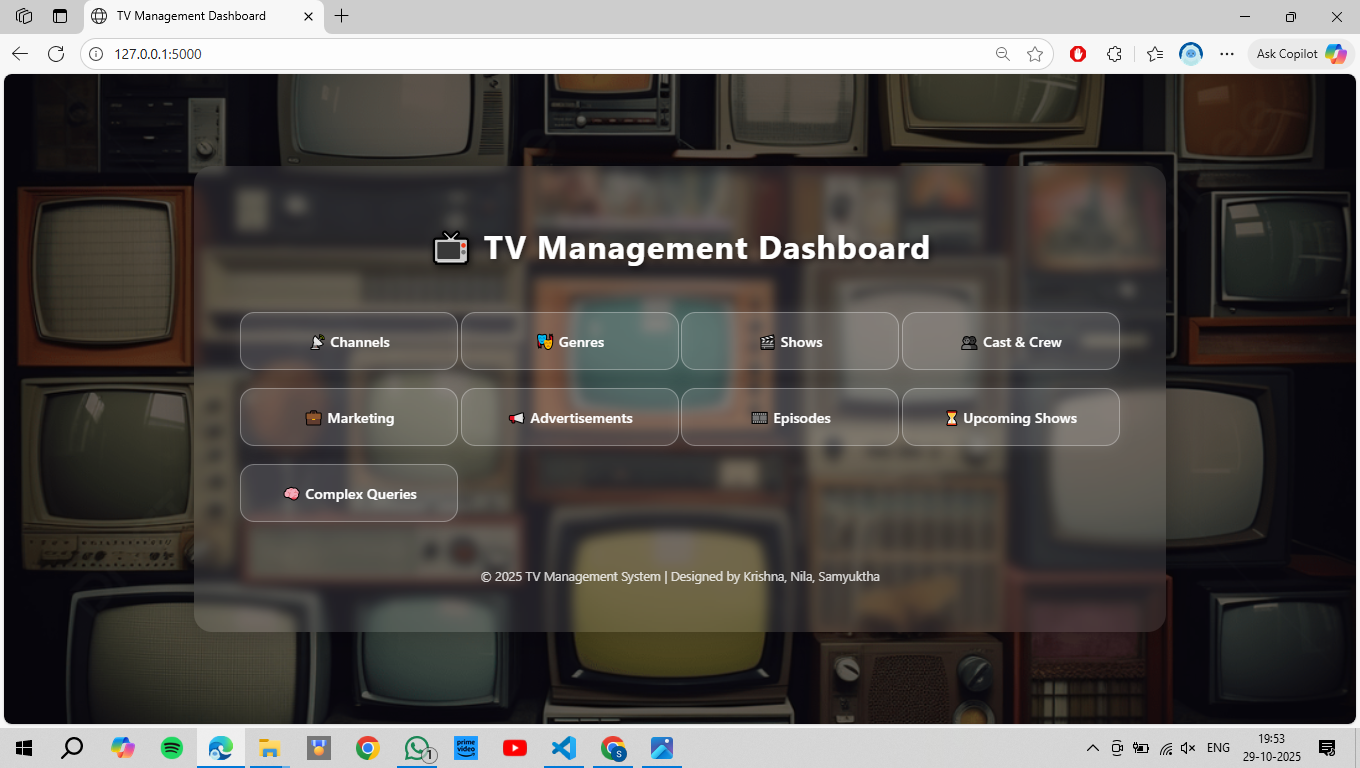
      <p>© 2025 TV Management System | Designed by Krishna, Nila, Samyuktha</p>

    </div>

  </div>

</body>

</html>



CHANNEL:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Channel Management</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      margin: 0;

      padding: 0;

      height: 100vh;

      background: url('channel.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

    }

    .overlay {

      background: rgba(0, 0, 0, 0.5);

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

      z-index: 0;

    }

    .container {

      position: relative;

      z-index: 1;

      width: 90%;

      max-width: 950px;

      background: transparent;

      border-radius: 16px;

      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);

      padding: 30px 40px;

      backdrop-filter: blur(3px);

      border: 1px solid rgba(255, 255, 255, 0.3);

    }

    h2 {

      text-align: center;

      color: white;

      text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);

      margin-bottom: 25px;

      font-size: 28px;

      letter-spacing: 1px;

    }

    form[action="/channels/add"] {

      display: grid;

      grid-template-columns: 1fr 1fr;

      gap: 15px;

      margin-bottom: 25px;

    }

    input {

      padding: 10px;

      border: 1px solid #ccc;

      border-radius: 8px;

      background: rgba(255, 255, 255, 0.8);

      transition: border-color 0.3s;

    }

    input:focus {

      border-color: gray;

      outline: none;

    }

    button {

      background: papayawhip;

      color: black;

      border: none;

      border-radius: 8px;

      padding: 10px 14px;

      cursor: pointer;

      transition: background 0.3s;

    }

    button:hover {

      background: peachpuff;

    }

    table {

      width: 100%;

      border-collapse: collapse;

      text-align: center;

      margin-top: 15px;

      color: #2d3436;

      background: rgba(255, 255, 255, 0.7);

      border-radius: 10px;

      overflow: hidden;

    }

    th, td {

      padding: 10px;

      border-bottom: 1px solid rgba(0, 0, 0, 0.1);

    }

    th {

      background-color: papayawhip;

      color: black;

    }

    tr:hover {

      background-color: rgba(200, 200, 255, 0.3);

      transition: 0.3s;

    }

    a {

      color: papayawhip;

      text-decoration: none;

      transition: color 0.3s;

    }

    a:hover {

      color: peachpuff;

    }

    .back {

      display: inline-block;

      margin-top: 20px;

      text-decoration: none;

      background-color: papayawhip;

      color: black;

      padding: 10px 18px;

      border-radius: 8px;

      transition: background-color 0.3s;

    }

    .back:hover {

      background-color: honeydew;

    }

    table input {

      width: 100%;

      border: 1px solid #ccc;

      border-radius: 5px;

      padding: 6px;

      background: rgba(255, 255, 255, 0.8);

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h2>Channel Management</h2>

    <form action="/channels/add" method="post">

      <input name="channel\_id" placeholder="Channel ID" required>

      <input name="channel\_name" placeholder="Channel Name" required>

      <input name="satellite" placeholder="Satellite">

      <input name="num\_of\_shows" type="number" placeholder="Number of Shows">

      <input name="location" placeholder="Location">

      <button type="submit">Add Channel</button>

    </form>

    <table>

      <tr>

        <th>ID</th>

        <th>Name</th>

        <th>Satellite</th>

        <th>#Shows</th>

        <th>Location</th>

        <th>Actions</th>

      </tr>

      {% for c in channels %}

      <tr>

        <form action="/channels/update/{{ c.channel\_id }}" method="post">

          <td>{{ c.channel\_id }}</td>

          <td><input name="channel\_name" value="{{ c.channel\_name }}"></td>

          <td><input name="satellite" value="{{ c.satellite }}"></td>

          <td><input name="num\_of\_shows" value="{{ c.num\_of\_shows }}"></td>

          <td><input name="location" value="{{ c.location }}"></td>

          <td>

            <button type="submit">Update</button>

            <a href="/channels/delete/{{ c.channel\_id }}">🗑 Delete</a>

          </td>

        </form>

      </tr>

      {% endfor %}

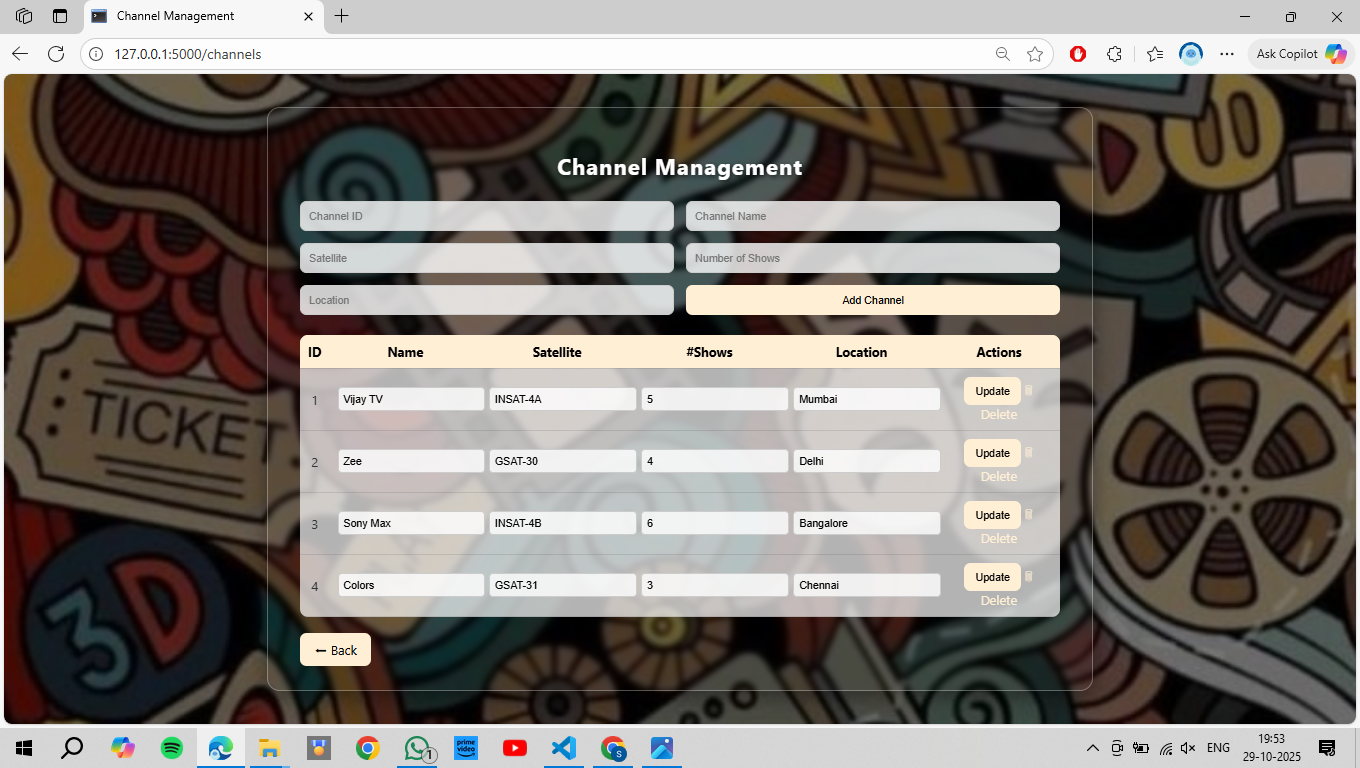
    </table>

    <a class="back" href="/">⬅ Back</a>

  </div>

</body>

</html>



SHOW:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Show Management</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      margin: 0;

      padding: 0;

      height: 100vh;

      background: url('show.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

    }

    .overlay {

      background: rgba(0, 0, 0, 0.5);

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

      z-index: 0;

    }

    .container {

      position: relative;

      z-index: 1;

      width: 90%;

      max-width: 1000px;

      background: transparent;

      border-radius: 16px;

      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);

      padding: 30px 40px;

      backdrop-filter: blur(3px);

      border: 1px solid rgba(255, 255, 255, 0.3);

    }

    h2 {

      text-align: center;

      color: white;

      text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);

      margin-bottom: 25px;

      font-size: 28px;

      letter-spacing: 1px;

    }

    form[action="/shows/add"] {

      display: grid;

      grid-template-columns: repeat(3, 1fr);

      gap: 12px;

      margin-bottom: 25px;

    }

    input {

      padding: 10px;

      border: 1px solid #ccc;

      border-radius: 8px;

      background: rgba(255, 255, 255, 0.8);

      transition: border-color 0.3s;

    }

    input:focus {

      border-color: gray;

      outline: none;

    }

    button {

      background: gainsboro;

      color: black;

      border: none;

      border-radius: 8px;

      padding: 10px 14px;

      cursor: pointer;

      transition: background 0.3s;

    }

    button:hover {

      background: gray;

    }

    table {

      width: 100%;

      border-collapse: collapse;

      text-align: center;

      margin-top: 15px;

      color: #2d3436;

      background: rgba(255, 255, 255, 0.7);

      border-radius: 10px;

      overflow: hidden;

    }

    th, td {

      padding: 10px;

      border-bottom: 1px solid rgba(0, 0, 0, 0.1);

    }

    th {

      background-color: gainsboro;

      color: black;

    }

    tr:hover {

      background-color: rgba(200, 200, 255, 0.3);

      transition: 0.3s;

    }

    a {

      color: black;

      text-decoration: none;

      transition: color 0.3s;

    }

    a:hover {

      color: gray;

    }

    .back {

      display: inline-block;

      margin-top: 20px;

      text-decoration: none;

      background-color: gainsboro;

      color: black;

      padding: 10px 18px;

      border-radius: 8px;

      transition: background-color 0.3s;

    }

    .back:hover {

      background-color: gray;

    }

    table input {

      width: 100%;

      border: 1px solid #ccc;

      border-radius: 5px;

      padding: 6px;

      background: rgba(255, 255, 255, 0.8);

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h2>Show Management</h2>

    <form action="/shows/add" method="post">

      <input name="show\_id" placeholder="Show ID" type="number" required>

      <input name="channel\_id" placeholder="Channel ID" type="number" required>

      <input name="genre\_id" placeholder="Genre ID" type="number" required>

      <input name="title" placeholder="Title" required>

      <input name="air\_time" placeholder="Air Time" type="time" required>

      <input name="director\_id" placeholder="Director ID" type="number" required>

      <button type="submit">Add Show</button>

    </form>

    <table>

      <tr>

        <th>ID</th>

        <th>Channel</th>

        <th>Genre</th>

        <th>Title</th>

        <th>Air Time</th>

        <th>Director</th>

        <th>Actions</th>

      </tr>

      {% for s in shows %}

      <tr>

        <form action="/shows/update/{{ s.show\_id }}" method="post">

          <td>{{ s.show\_id }}</td>

          <td><input name="channel\_id" value="{{ s.channel\_id }}"></td>

          <td><input name="genre\_id" value="{{ s.genre\_id }}"></td>

          <td><input name="title" value="{{ s.title }}"></td>

          <td><input name="air\_time" value="{{ s.air\_time }}"></td>

          <td><input name="director\_id" value="{{ s.director\_id }}"></td>

          <td>

            <button type="submit">Update</button>

            <a href="/shows/delete/{{ s.show\_id }}">🗑 Delete</a>

          </td>

        </form>

      </tr>

      {% endfor %}

    </table>

    <a class="back" href="/">⬅ Back</a>

  </div>

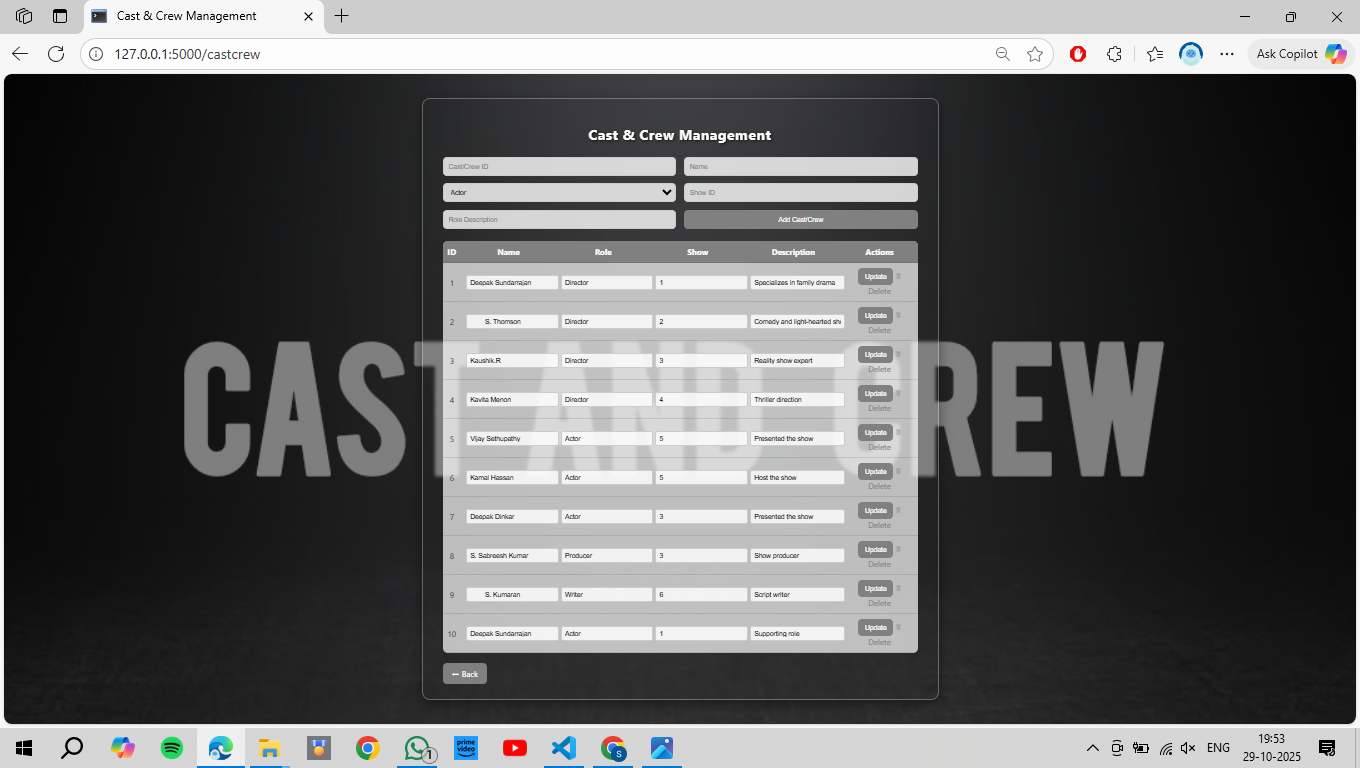
</body>

</html>



CAST AND CREW:

<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>Cast & Crew Management</title> <style> body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; margin: 0; padding: 0; height: 100vh; background: url('cc.jpg') no-repeat center center fixed; background-size: cover; display: flex; justify-content: center; align-items: center; } .overlay { background: rgba(0, 0, 0, 0.5); position: fixed; top: 0; left: 0; width: 100%; height: 100%; z-index: 0; } /\* 🌈 Frosted Glass Container \*/ .container { position: relative; z-index: 1; width: 90%; max-width: 950px; background: transparent; /\* transparent glass \*/ border-radius: 16px; box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4); padding: 30px 40px; backdrop-filter: blur(3px); border: 1px solid rgba(255, 255, 255, 0.3); } /\* 🏷️ Title \*/ h2 { text-align: center; color: white; text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7); margin-bottom: 25px; font-size: 28px; letter-spacing: 1px; } /\* 📋 Add Form \*/ form[action="/castcrew/add"] { display: grid; grid-template-columns: 1fr 1fr; gap: 15px; margin-bottom: 25px; } input, select { padding: 10px; border: 1px solid #ccc; border-radius: 8px; transition: border-color 0.3s; background: rgba(255, 255, 255, 0.8); } input:focus, select:focus { border-color: gray; outline: none; } button { background: gray; color: white; border: none; border-radius: 8px; padding: 10px 14px; cursor: pointer; transition: background 0.3s; } button:hover { background: grey; } /\* 📊 Table Styling \*/ table { width: 100%; border-collapse: collapse; text-align: center; margin-top: 15px; color: #2d3436; background: rgba(255, 255, 255, 0.7); border-radius: 10px; overflow: hidden; } th, td { padding: 10px; border-bottom: 1px solid rgba(0, 0, 0, 0.1); } th { background-color: gray; color: white; } tr:hover { background-color: rgba(200, 200, 255, 0.3); transition: 0.3s; } a { color: gray; text-decoration: none; transition: color 0.3s; } a:hover { color: grey; } .back { display: inline-block; margin-top: 20px; text-decoration: none; background-color: gray; color: white; padding: 10px 18px; border-radius: 8px; transition: background-color 0.3s; } .back:hover { background-color: grey; } /\* 📝 Update form inside table \*/ table input { width: 100%; border: 1px solid #ccc; border-radius: 5px; padding: 6px; background: rgba(255, 255, 255, 0.8); } </style> </head> <body> <div class="overlay"></div> <div class="container"> <h2>Cast & Crew Management</h2> <form action="/castcrew/add" method="post"> <input name="castcrew\_id" placeholder="Cast/Crew ID" required> <input name="name" placeholder="Name" required> <select name="role\_type"> <option>Actor</option> <option>Director</option> <option>Producer</option> <option>Writer</option> </select> <input name="show\_id" type="number" placeholder="Show ID"> <input name="role\_description" placeholder="Role Description"> <button type="submit">Add Cast/Crew</button> </form> <table> <tr> <th>ID</th> <th>Name</th> <th>Role</th> <th>Show</th> <th>Description</th> <th>Actions</th> </tr> {% for p in castcrew %} <tr> <form action="/castcrew/update/{{ p.castcrew\_id }}" method="post"> <td>{{ p.castcrew\_id }}</td> <td><input name="name" value="{{ p.name }}"></td> <td><input name="role\_type" value="{{ p.role\_type }}"></td> <td><input name="show\_id" value="{{ p.show\_id }}"></td> <td><input name="role\_description" value="{{ p.role\_description }}"></td> <td> <button type="submit">Update</button> <a href="/castcrew/delete/{{ p.castcrew\_id }}">🗑 Delete</a> </td> </form> </tr> {% endfor %} </table> <a class="back" href="/">⬅ Back</a> </div> </body> </html>



GENRE:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Genre Management</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      margin: 0;

      padding: 0;

      height: 100vh;

      background: url('genre.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

    }

    .overlay {

      background: rgba(0, 0, 0, 0.5);

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

      z-index: 0;

    }

    .container {

      position: relative;

      z-index: 1;

      width: 90%;

      max-width: 800px;

      background: transparent;

      border-radius: 16px;

      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);

      padding: 30px 40px;

      backdrop-filter: blur(3px);

      border: 1px solid rgba(255, 255, 255, 0.3);

    }

    h2 {

      text-align: center;

      color: white;

      text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);

      margin-bottom: 25px;

      font-size: 28px;

      letter-spacing: 1px;

    }

    form[action="/genres/add"] {

      display: grid;

      grid-template-columns: 1fr 1fr;

      gap: 15px;

      margin-bottom: 25px;

    }

    input {

      padding: 10px;

      border: 1px solid #ccc;

      border-radius: 8px;

      background: rgba(255, 255, 255, 0.8);

      transition: border-color 0.3s;

    }

    input:focus {

      border-color: gray;

      outline: none;

    }

    button {

      background: gainsboro;

      color: black;

      border: none;

      border-radius: 8px;

      padding: 10px 14px;

      cursor: pointer;

      transition: background 0.3s;

    }

    button:hover {

      background: gray;

    }

    table {

      width: 100%;

      border-collapse: collapse;

      text-align: center;

      margin-top: 15px;

      color: #2d3436;

      background: rgba(255, 255, 255, 0.7);

      border-radius: 10px;

      overflow: hidden;

    }

    th, td {

      padding: 10px;

      border-bottom: 1px solid rgba(0, 0, 0, 0.1);

    }

    th {

      background-color: gainsboro;

      color: black;

    }

    tr:hover {

      background-color: rgba(200, 200, 255, 0.3);

      transition: 0.3s;

    }

    a {

      color: black;

      text-decoration: none;

      transition: color 0.3s;

    }

    a:hover {

      color: gray;

    }

    .back {

      display: inline-block;

      margin-top: 20px;

      text-decoration: none;

      background-color: gainsboro;

      color: black;

      padding: 10px 18px;

      border-radius: 8px;

      transition: background-color 0.3s;

    }

    .back:hover {

      background-color: gray;

    }

    table input {

      width: 100%;

      border: 1px solid #ccc;

      border-radius: 5px;

      padding: 6px;

      background: rgba(255, 255, 255, 0.8);

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h2>Genre Management</h2>

    <form action="/genres/add" method="post">

      <input name="genre\_id" placeholder="Genre ID" required>

      <input name="genre\_name" placeholder="Genre Name" required>

      <button type="submit">Add Genre</button>

    </form>

    <table>

      <tr>

        <th>ID</th>

        <th>Name</th>

        <th>Actions</th>

      </tr>

      {% for g in genres %}

      <tr>

        <form action="/genres/update/{{ g.genre\_id }}" method="post">

          <td>{{ g.genre\_id }}</td>

          <td><input name="genre\_name" value="{{ g.genre\_name }}"></td>

          <td>

            <button type="submit">Update</button>

            <a href="/genres/delete/{{ g.genre\_id }}">🗑 Delete</a>

          </td>

        </form>

      </tr>

      {% endfor %}

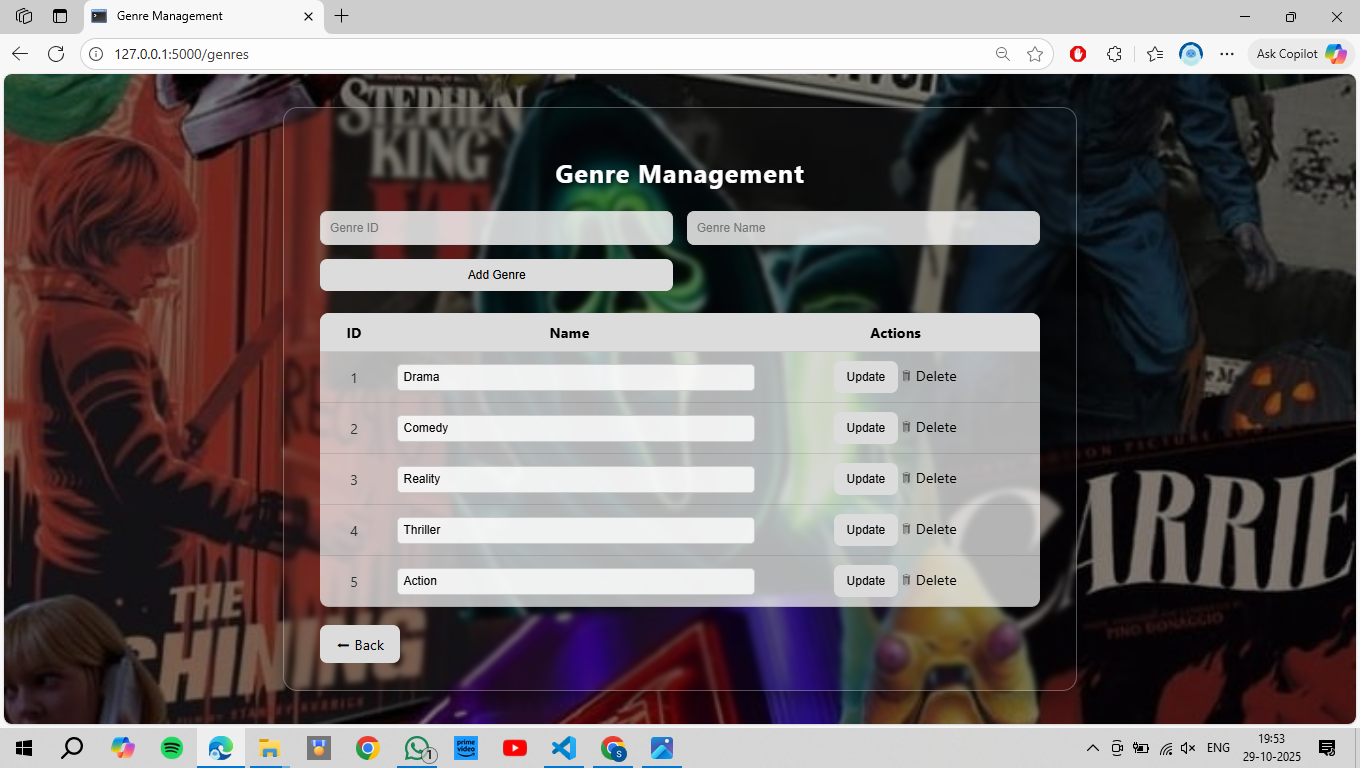
    </table>

    <a class="back" href="/">⬅ Back</a>

  </div>

</body>

</html>



EPISODES:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Episode Management</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      margin: 0;

      padding: 0;

      height: 100vh;

      background: url('epi.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

    }

    .overlay {

      background: rgba(0, 0, 0, 0.5);

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

      z-index: 0;

    }

    .container {

      position: relative;

      z-index: 1;

      width: 90%;

      max-width: 950px;

      background: transparent;

      border-radius: 16px;

      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);

      padding: 30px 40px;

      backdrop-filter: blur(3px);

      border: 1px solid rgba(255, 255, 255, 0.3);

    }

    h2 {

      text-align: center;

      color: white;

      text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);

      margin-bottom: 25px;

      font-size: 28px;

      letter-spacing: 1px;

    }

    form[action="/episodes/add"] {

      display: grid;

      grid-template-columns: 1fr 1fr;

      gap: 15px;

      margin-bottom: 25px;

    }

    input {

      padding: 10px;

      border: 1px solid #ccc;

      border-radius: 8px;

      background: rgba(255, 255, 255, 0.8);

      transition: border-color 0.3s;

    }

    input:focus {

      border-color: gray;

      outline: none;

    }

    button {

      background: gainsboro;

      color: black;

      border: none;

      border-radius: 8px;

      padding: 10px 14px;

      cursor: pointer;

      transition: background 0.3s;

    }

    button:hover {

      background: grey;

    }

    table {

      width: 100%;

      border-collapse: collapse;

      text-align: center;

      margin-top: 15px;

      color: #2d3436;

      background: rgba(255, 255, 255, 0.7);

      border-radius: 10px;

      overflow: hidden;

    }

    th, td {

      padding: 10px;

      border-bottom: 1px solid rgba(0, 0, 0, 0.1);

    }

    th {

      background-color: gainsboro;

      color: black;

    }

    tr:hover {

      background-color: rgba(200, 200, 255, 0.3);

      transition: 0.3s;

    }

    a {

      color: black;

      text-decoration: none;

      transition: color 0.3s;

    }

    a:hover {

      color: grey;

    }

    .back {

      display: inline-block;

      margin-top: 20px;

      text-decoration: none;

      background-color: gainsboro;

      color: black;

      padding: 10px 18px;

      border-radius: 8px;

      transition: background-color 0.3s;

    }

    .back:hover {

      background-color: grey;

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h2>Episode Management</h2>

    <form action="/episodes/add" method="post">

      <input name="episode\_id" placeholder="Episode ID" required>

      <input name="show\_id" type="number" placeholder="Show ID">

      <input name="season\_number" type="number" placeholder="Season #">

      <input name="episode\_number" type="number" placeholder="Episode #">

      <input name="duration" placeholder="Duration (e.g. 00:30:00)">

      <input name="air\_date" type="date" placeholder="Air Date">

      <button type="submit">Add Episode</button>

    </form>

    <table>

      <tr>

        <th>ID</th>

        <th>Show</th>

        <th>Season</th>

        <th>Episode</th>

        <th>Duration</th>

        <th>Air Date</th>

        <th>Actions</th>

      </tr>

      {% for e in episodes %}

      <tr>

        <td>{{ e.episode\_id }}</td>

        <td>{{ e.show\_id }}</td>

        <td>{{ e.season\_number }}</td>

        <td>{{ e.episode\_number }}</td>

        <td>{{ e.duration }}</td>

        <td>{{ e.air\_date }}</td>

        <td><a href="/episodes/delete/{{ e.episode\_id }}">🗑 Delete</a></td>

      </tr>

      {% endfor %}

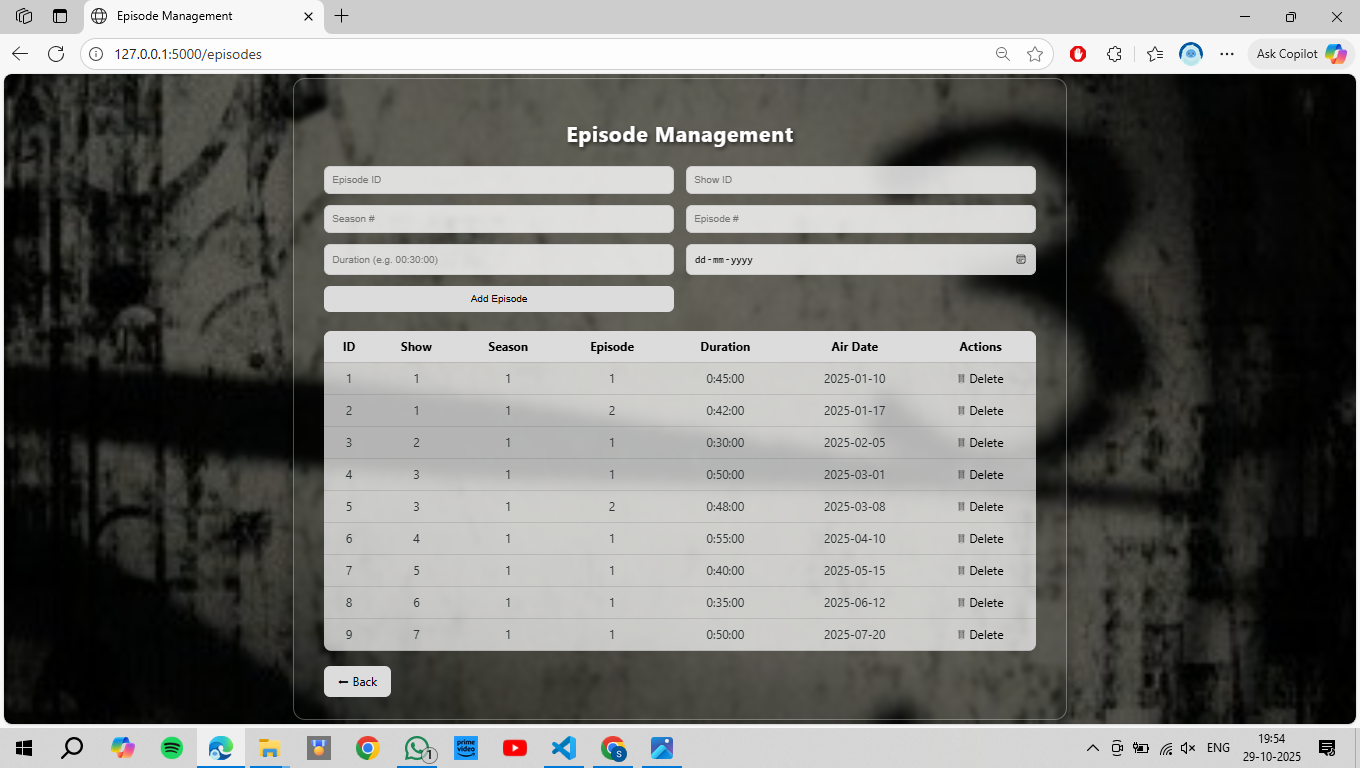
    </table>

    <a class="back" href="/">⬅ Back</a>

  </div>

</body>

</html>



UPCOMING:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Upcoming Shows</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      margin: 0;

      padding: 0;

      height: 100vh;

      background: url('upcoming.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

    }

    .overlay {

      background: rgba(0, 0, 0, 0.5);

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

      z-index: 0;

    }

    .container {

      position: relative;

      z-index: 1;

      width: 90%;

      max-width: 900px;

      background: transparent;

      border-radius: 16px;

      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);

      padding: 30px 40px;

      backdrop-filter: blur(8px);

      border: 1px solid rgba(255, 255, 255, 0.3);

    }

    h2 {

      text-align: center;

      color: white;

      text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);

      margin-bottom: 25px;

      font-size: 28px;

      letter-spacing: 1px;

    }

    form {

      display: grid;

      grid-template-columns: repeat(auto-fit, minmax(180px, 1fr));

      gap: 15px;

      margin-bottom: 25px;

    }

    input {

      padding: 10px;

      border: 1px solid #ccc;

      border-radius: 8px;

      background: rgba(255, 255, 255, 0.8);

      transition: border-color 0.3s;

    }

    input:focus {

      border-color: gray;

      outline: none;

    }

    button {

      background: gainsboro;

      color: black;

      border: none;

      border-radius: 8px;

      padding: 10px 14px;

      cursor: pointer;

      transition: background 0.3s;

    }

    button:hover {

      background: gray;

      color: white;

    }

    table {

      width: 100%;

      border-collapse: collapse;

      text-align: center;

      margin-top: 15px;

      color: #2d3436;

      background: rgba(255, 255, 255, 0.7);

      border-radius: 10px;

      overflow: hidden;

    }

    th, td {

      padding: 10px;

      border-bottom: 1px solid rgba(0, 0, 0, 0.1);

    }

    th {

      background-color: gainsboro;

      color: black;

    }

    tr:hover {

      background-color: rgba(200, 200, 255, 0.3);

      transition: 0.3s;

    }

    a {

      color: black;

      text-decoration: none;

      transition: color 0.3s;

    }

    a:hover {

      color: gray;

    }

    .back {

      display: inline-block;

      margin-top: 20px;

      text-decoration: none;

      background-color: gainsboro;

      color: black;

      padding: 10px 18px;

      border-radius: 8px;

      transition: background-color 0.3s;

    }

    .back:hover {

      background-color: gray;

      color: white;

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h2>Upcoming Shows</h2>

    <form action="/upcoming/add" method="post">

      <input name="upcoming\_id" type="number" placeholder="Upcoming ID" required>

      <input name="channel\_id" type="number" placeholder="Channel ID" required>

      <input name="title" placeholder="Title" required>

      <input name="replacement\_show\_id" type="number" placeholder="Replacement Show ID">

      <input name="director\_id" type="number" placeholder="Director ID">

      <button type="submit">Add</button>

    </form>

    <table>

      <tr>

        <th>ID</th>

        <th>Channel</th>

        <th>Title</th>

        <th>Replacement</th>

        <th>Director</th>

        <th>Actions</th>

      </tr>

      {% for u in upcoming %}

      <tr>

        <td>{{ u.upcoming\_id }}</td>

        <td>{{ u.channel\_name }}</td>

        <td>{{ u.title }}</td>

        <td>{{ u.replacement\_show\_id }}</td>

        <td>{{ u.director\_id }}</td>

        <td><a href="/upcoming/promote/{{ u.upcoming\_id }}">Promote</a></td>

      </tr>

      {% endfor %}

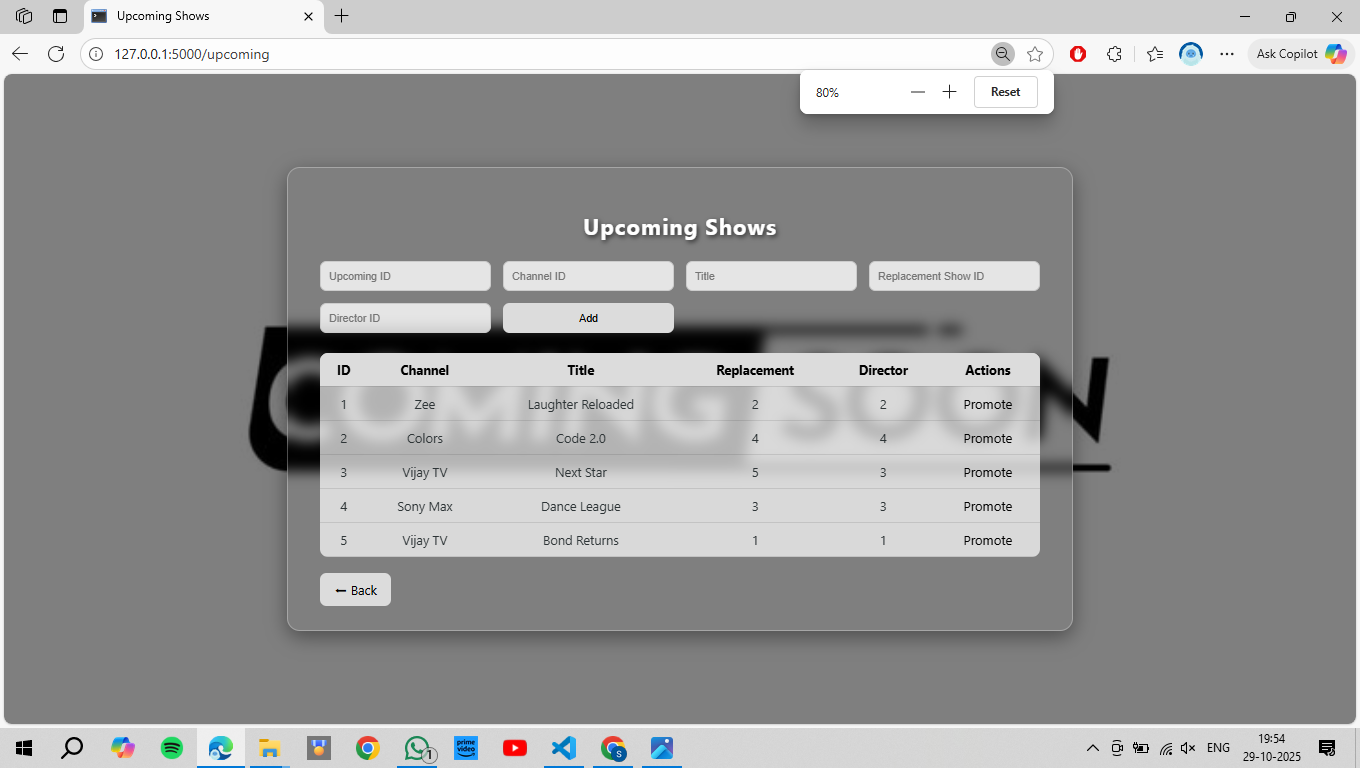
    </table>

    <a class="back" href="/">⬅ Back</a>

  </div>

</body>

</html>



MARKETING:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Marketing Management</title>

  <style>

    body {

      margin: 0;

      padding: 0;

      height: 100vh;

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      background: url('market.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

    }

    .overlay {

      position: fixed;

      top: 0; left: 0;

      width: 100%; height: 100%;

      background: rgba(0, 0, 0, 0.55);

      z-index: 0;

    }

    .container {

      position: relative;

      z-index: 1;

      width: 100%;

      max-width: 10000px;

      background: rgba(255, 255, 255, 0.1);

      backdrop-filter: blur(8px);

      border: 1px solid rgba(255, 255, 255, 0.3);

      border-radius: 16px;

      padding: 35px 45px;

      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);

    }

    h2 {

      text-align: center;

      color: #fff;

      text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.6);

      font-size: 30px;

      letter-spacing: 1px;

      margin-bottom: 25px;

    }

    form[action="/marketing/add"] {

      display: grid;

      grid-template-columns: repeat(6, 1fr);

      gap: 12px;

      margin-bottom: 25px;

    }

    input {

      padding: 10px;

      border: 1px solid #ccc;

      border-radius: 8px;

      background: rgba(255, 255, 255, 0.85);

      font-size: 14px;

      transition: 0.3s;

    }

    input:focus {

      outline: none;

      border-color: gray;

    }

    button {

      background: gainsboro;

      color: black;

      border: none;

      border-radius: 8px;

      padding: 10px 14px;

      cursor: pointer;

      font-weight: bold;

      transition: 0.3s;

    }

    button:hover {

      background: gray;

      color: white;

    }

    table {

      width: 100%;

      border-collapse: collapse;

      background: rgba(255, 255, 255, 0.8);

      border-radius: 10px;

      overflow: hidden;

      text-align: center;

    }

    th, td {

      padding: 10px;

      border-bottom: 1px solid rgba(0, 0, 0, 0.1);

    }

    th {

      background-color: gainsboro;

      color: black;

      font-weight: bold;

    }

    tr:hover {

      background-color: rgba(200, 200, 255, 0.3);

      transition: 0.3s;

    }

    a {

      color: black;

      text-decoration: none;

      transition: 0.3s;

      font-weight: 500;

    }

    a:hover {

      color: gray;

    }

    .back {

      display: inline-block;

      margin-top: 20px;

      background-color: gainsboro;

      color: black;

      padding: 10px 18px;

      border-radius: 8px;

      text-decoration: none;

      transition: background 0.3s;

      font-weight: bold;

    }

    .back:hover {

      background-color: gray;

      color: white;

    }

    @media (max-width: 768px) {

      form[action="/marketing/add"] {

        grid-template-columns: 1fr;

      }

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h2>Marketing Management</h2>

    <form action="/marketing/add" method="post">

      <input name="marketing\_id" placeholder="Marketing ID" required>

      <input name="channel\_id" placeholder="Channel ID" type="number" required>

      <input name="show\_id" placeholder="Show ID" type="number" required>

      <input name="trp" placeholder="TRP" type="number" step="0.01" required>

      <input name="cost\_of\_production" placeholder="Cost of Production" type="number" step="0.01" required>

      <button type="submit">Add</button>

    </form>

    <table>

      <tr>

        <th>ID</th>

        <th>Channel</th>

        <th>Show</th>

        <th>TRP</th>

        <th>Cost</th>

        <th>Actions</th>

      </tr>

      {% for m in marketing %}

      <tr>

        <td>{{ m.marketing\_id }}</td>

        <td>{{ m.channel\_id }} - {{ m.channel\_name }}</td>

        <td>{{ m.show\_id }} - {{ m.title }}</td>

        <td>{{ m.trp }}</td>

        <td>{{ m.cost\_of\_production }}</td>

        <td><a href="/marketing/delete/{{ m.marketing\_id }}">🗑 Delete</a></td>

      </tr>

      {% endfor %}

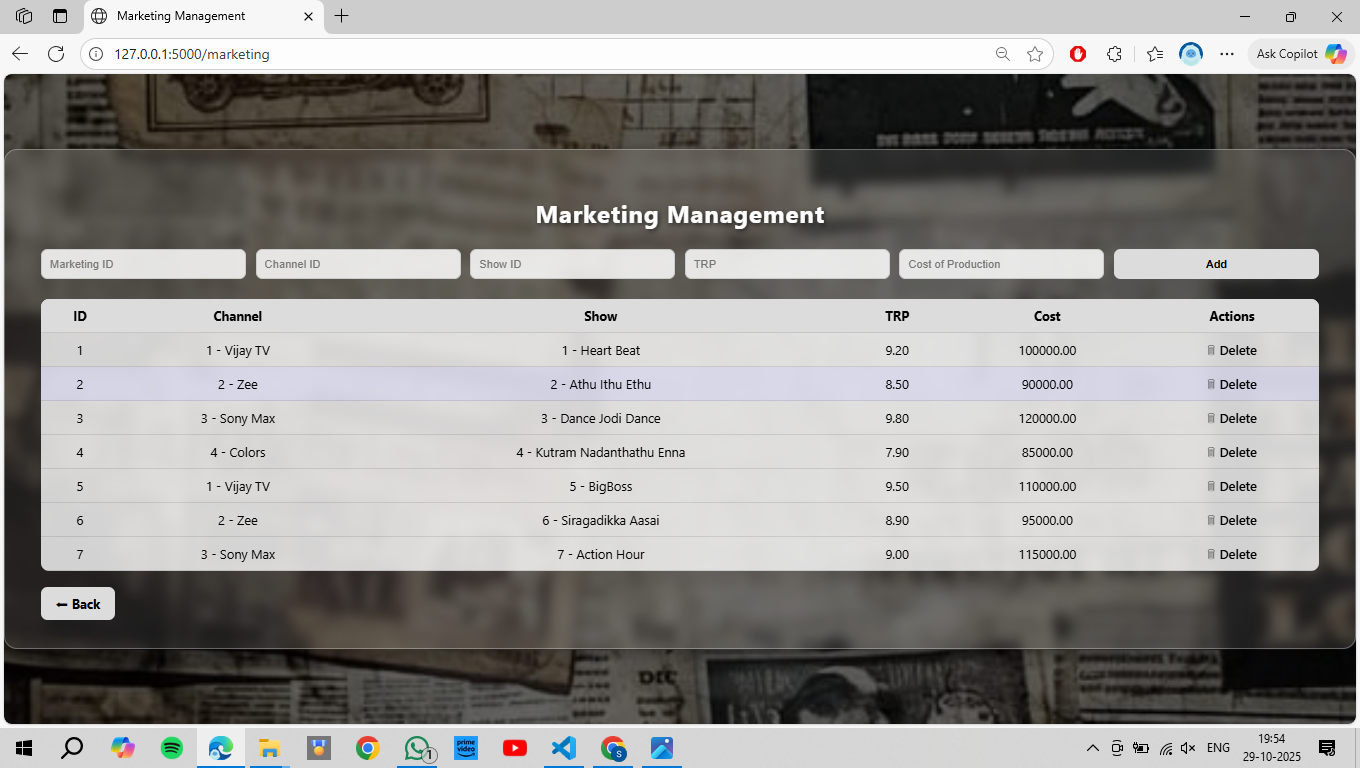
    </table>

    <a class="back" href="/">⬅ Back</a>

  </div>

</body>

</html>



ADVERTISEMENT:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Advertisement Management</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      margin: 0;

      padding: 0;

      height: 100vh;

      background: url('ads.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

      color:white;

    }

    .overlay {

      background: rgba(0, 0, 0, 0.5);

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

      z-index: 0;

    }

    .container {

  position: relative;

  z-index: 1;

  width: 90%;

  max-width: 950px;

  background: transparent; /\* translucent \*/

  border-radius: 16px;

  box-shadow: 0 8px 25px rgba(0, 0, 0, 0.3);

  padding: 30px 40px;

  backdrop-filter: blur(3px);  /\* frosted glass effect \*/

}

    h2 {

      text-align: center;

      color: white;

      margin-bottom: 25px;

    }

    form {

      display: grid;

      grid-template-columns: 1fr 1fr;

      gap: 15px;

      margin-bottom: 25px;

    }

    input {

      padding: 10px;

      border: 1px solid #ccc;

      border-radius: 8px;

      transition: border-color 0.3s;

    }

    input:focus {

      border-color: palevioletred;

      outline: none;

    }

    button {

      grid-column: 1 / -1;

      background: plum;

      color: white;

      border: solid;

      border-radius: 8px;

      padding: 12px;

      font-size: 16px;

      cursor: pointer;

      transition: background 0.3s;

    }

    button:hover {

      background: palevioletred;

    }

    table {

      width: 100%;

      border-collapse: collapse;

      margin-top: 20px;

      text-align: center;

    }

    th, td {

      padding: 12px;

      border-bottom: 1px solid #ddd;

    }

    th {

      background-color: plum;

      color: white;

    }

    tr:hover {

      background-color: #f1f2f6;

      transition: 0.3s;

    }

    a {

      color: plum;

      text-decoration: none;

      transition: color 0.3s;

    }

    a:hover {

      color: palevioletred;

    }

    .back {

      display: inline-block;

      margin-top: 20px;

      text-decoration: none;

      background-color: plum;

      color: white;

      padding: 10px 18px;

      border-radius: 8px;

      transition: background-color 0.3s;

    }

    .back:hover {

      background-color: palevioletred;

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h2>Advertisement Management</h2>

    <form action="/ads/add" method="post">

      <input name="ad\_id" placeholder="Ad ID" required>

      <input name="channel\_id" type="number" placeholder="Channel ID">

      <input name="company\_name" placeholder="Company Name">

      <input name="num\_of\_ads" type="number" placeholder="Number of Ads">

      <input name="duration" placeholder="Duration (e.g. 00:01:30)">

      <button type="submit">Add Advertisement</button>

    </form>

    <table>

      <tr>

        <th>ID</th>

        <th>Channel</th>

        <th>Company</th>

        <th>No. of Ads</th>

        <th>Duration</th>

        <th>Actions</th>

      </tr>

      {% for a in ads %}

      <tr>

        <td>{{ a.ad\_id }}</td>

        <td>{{ a.channel\_name }}</td>

        <td>{{ a.company\_name }}</td>

        <td>{{ a.num\_of\_ads }}</td>

        <td>{{ a.duration }}</td>

        <td><a href="/ads/delete/{{ a.ad\_id }}">🗑 Delete</a></td>

      </tr>

      {% endfor %}

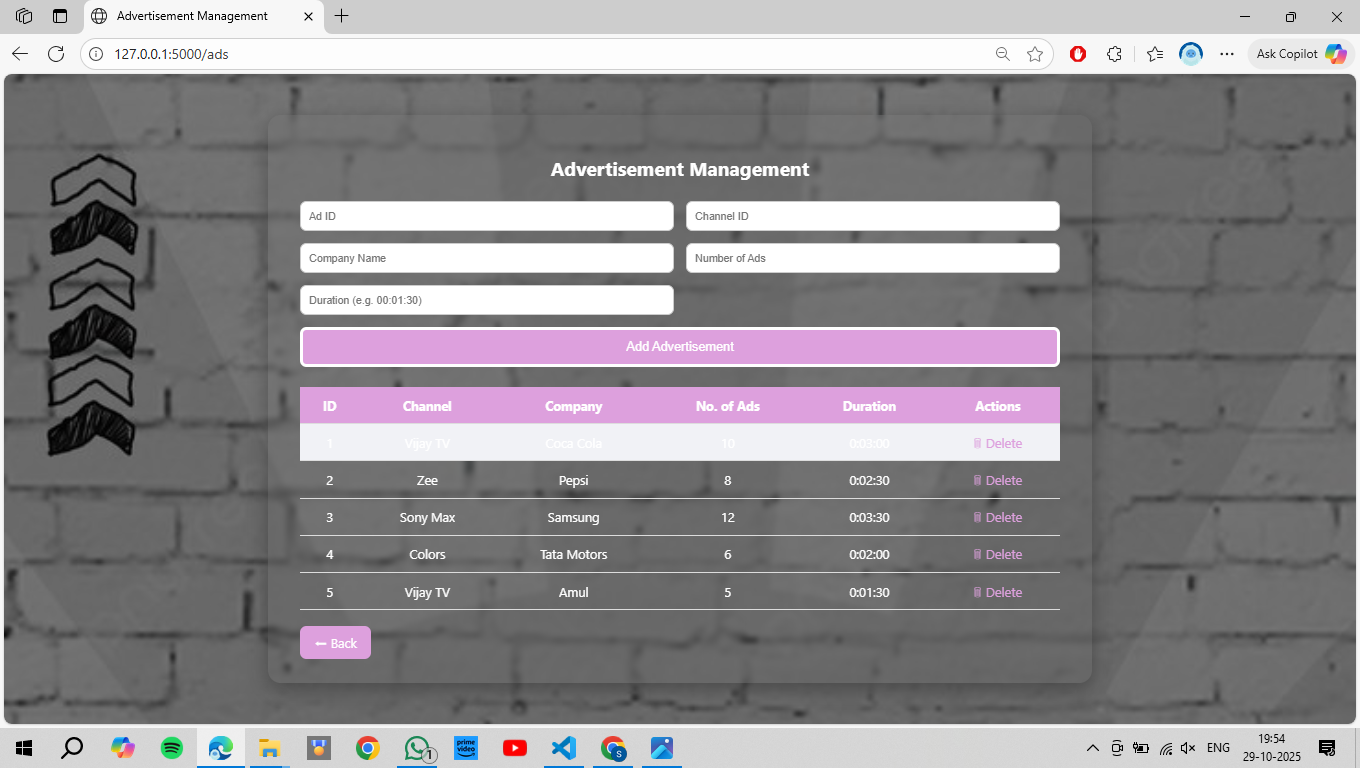
    </table>

    <a class="back" href="/">⬅ Back</a>

  </div>

</body>

</html>



QUERIES:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Query Dashboard</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      margin: 0;

      padding: 0;

      height: 100vh;

      background: url('query.jpg') no-repeat center center fixed;

      background-size: cover;

      display: flex;

      justify-content: center;

      align-items: center;

    }

    .overlay {

      background: rgba(0, 0, 0, 0.5);

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      height: 100%;

      z-index: 0;

    }

    .container {

      position: relative;

      z-index: 1;

      width: 90%;

      max-width: 950px;

      background: transparent;

      border-radius: 16px;

      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.4);

      padding: 40px;

      backdrop-filter: blur(3px);

      border: 1px solid rgba(255, 255, 255, 0.3);

      animation: fadeIn 0.8s ease-in-out;

    }

    @keyframes fadeIn {

      from { opacity: 0; transform: scale(0.95); }

      to { opacity: 1; transform: scale(1); }

    }

    h2 {

      text-align: center;

      color: white;

      text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.7);

      margin-bottom: 25px;

      font-size: 30px;

      letter-spacing: 1px;

    }

    form {

      display: flex;

      flex-wrap: wrap;

      justify-content: center;

      gap: 12px;

    }

    button {

      background: rgba(255, 255, 255, 0.8);

      color: black;

      border: none;

      border-radius: 10px;

      padding: 12px 18px;

      cursor: pointer;

      font-size: 15px;

      min-width: 260px;

      box-shadow: 0 3px 6px rgba(0,0,0,0.2);

      transition: all 0.3s ease-in-out;

      transform: scale(1);

    }

    button:hover {

      background: #dcdcdc;

      transform: scale(1.07);

      box-shadow: 0 5px 12px rgba(0,0,0,0.3);

    }

    button:active {

      transform: scale(0.97);

      background: gray;

      color: white;

    }

    .back {

      display: block;

      width: fit-content;

      margin: 25px auto 0;

      text-decoration: none;

      background-color: gainsboro;

      color: black;

      padding: 10px 20px;

      border-radius: 8px;

      transition: background-color 0.3s;

      text-align: center;

    }

    .back:hover {

      background-color: gray;

      color: white;

    }

  </style>

</head>

<body>

  <div class="overlay"></div>

  <div class="container">

    <h2>📊 Pre-defined Complex Queries</h2>

    <form action="/run\_named\_query" method="post">

      <button name="query\_name" value="avg\_trp\_per\_genre">Avg TRP per Genre</button>

      <button name="query\_name" value="top3\_expensive\_per\_channel">Top 3 Expensive Shows per Channel</button>

      <button name="query\_name" value="ad\_duration\_per\_channel">Ad Duration per Channel</button>

      <button name="query\_name" value="shows\_trp\_above9\_cost\_below\_avg">Shows TRP > 9 & Cost < Avg</button>

      <button name="query\_name" value="directors\_no\_show\_below5">Directors with No Show < 5 TRP</button>

      <button name="query\_name" value="channel\_biggest\_cost\_diff">Channel with Biggest Cost Diff</button>

      <button name="query\_name" value="channel\_summary\_view">Channel Summary View</button>

      <button name="query\_name" value="generate\_weekly\_schedule">Generate Weekly Schedule (Demo)</button>

      <button name="query\_name" value="shows\_short\_high\_trp">Short Duration & High TRP Shows</button>

      <button name="query\_name" value="trp\_above\_avg">TRP Above Average</button>

      <button name="query\_name" value="cost\_per\_genre\_channel">Cost per Genre Across Channels</button>

      <button name="query\_name" value="directors\_multiple\_genres">Directors with Multiple Genres</button>

      <button name="query\_name" value="directors\_also\_actors\_shows">Directors Who Are Also Actors</button>

      <button name="query\_name" value="avg\_episode\_duration\_per\_season">Avg Episode Duration per Season</button>

      <button name="query\_name" value="upcoming\_replacing\_low\_trp">Upcoming Replacing Low TRP Shows</button>

      <button name="query\_name" value="prevent\_trp\_trigger\_test">TRP Trigger Test (Will Error)</button>

      <button name="query\_name" value="add\_show\_with\_validation">Add Show with Validation (Demo)</button>

      <button name="query\_name" value="top\_3\_shows">Top 3 Shows</button>

      <button name="query\_name" value="avg\_trp\_per\_channel">Average TRP per Channel</button>

      <button name="query\_name" value="prime\_time\_shows">Prime Time Shows</button>

      <button name="query\_name" value="top\_director">Top Director</button>

      <button name="query\_name" value="top\_expensive\_channel">Top Expensive Channel</button>

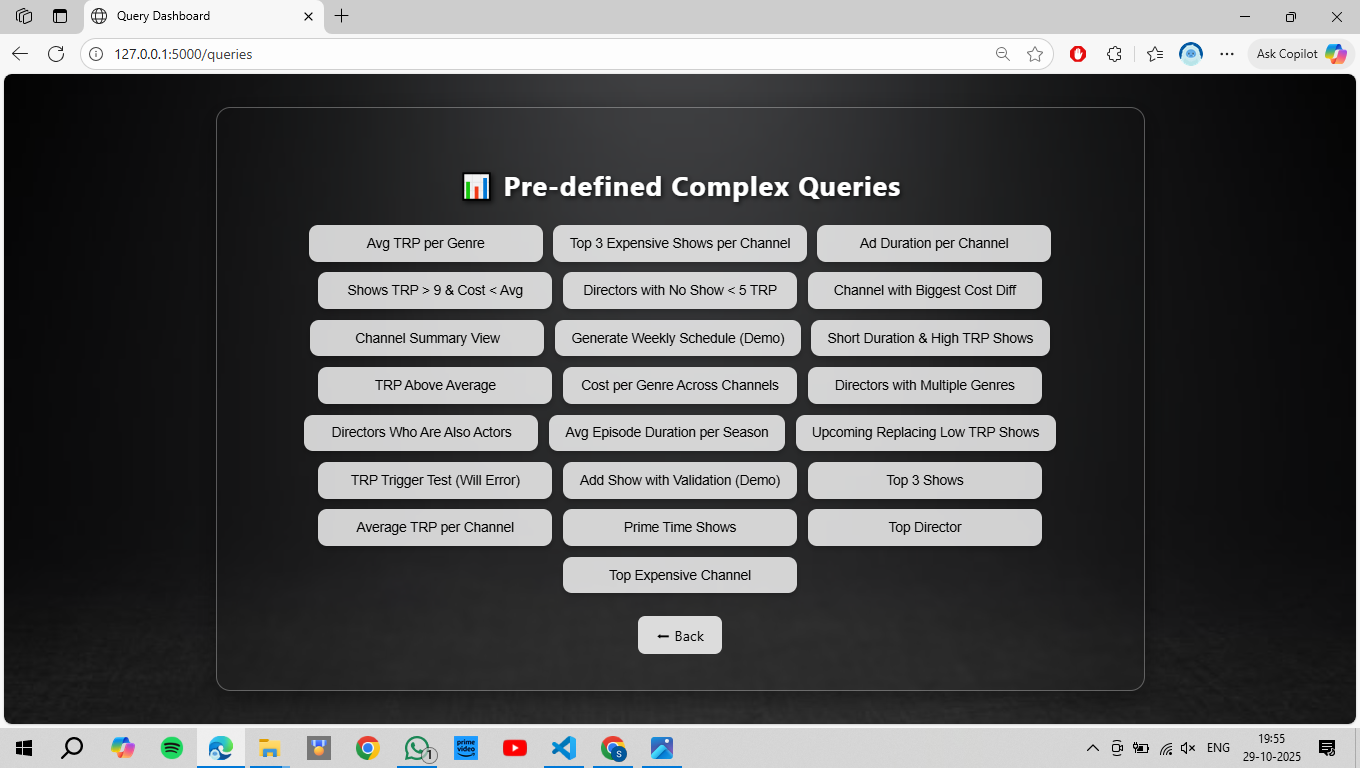
    </form>

    <a class="back" href="/">⬅ Back</a>

  </div>

</body>

</html>



QUERIES INSERTED:

-- 1. Add total\_duration to Show to track episodes sum (seconds using interval -> store as interval)

ALTER TABLE "Show"

ADD COLUMN IF NOT EXISTS total\_duration INTERVAL DEFAULT '00:00:00';

-- 2. Audit table: store deleted shows

CREATE TABLE IF NOT EXISTS deleted\_shows (

deleted\_id SERIAL PRIMARY KEY,

show\_id INT,

title VARCHAR(100),

deleted\_at TIMESTAMP DEFAULT now(),

deleted\_by VARCHAR(100),

reason TEXT

);

-- 3. VIEW: avg trp, total cost, total shows per channel (require Marketing & Show)

CREATE OR REPLACE VIEW channel\_summary AS

SELECT c.channel\_id, c.channel\_name,

ROUND(AVG(m.trp)::numeric,2) AS avg\_trp,

COALESCE(SUM(m.cost\_of\_production),0) AS total\_cost,

COALESCE(COUNT(DISTINCT s.show\_id),0) AS total\_shows

FROM "Channel" c

LEFT JOIN "Marketing" m ON c.channel\_id = m.channel\_id

LEFT JOIN "Show" s ON s.channel\_id = c.channel\_id

GROUP BY c.channel\_id, c.channel\_name;

-- 4. FUNCTION: store deleted show into deleted\_shows (to be used by trigger or manual delete)

CREATE OR REPLACE FUNCTION store\_deleted\_show()

RETURNS TRIGGER LANGUAGE plpgsql AS $$

BEGIN

INSERT INTO deleted\_shows(show\_id, title, deleted\_at, deleted\_by, reason)

VALUES (OLD.show\_id, OLD.title, now(), current\_user, TG\_ARGV[0]);

RETURN OLD;

END;

$$;

-- 5. TRIGGER: on delete from Show store into deleted\_shows automatically

DROP TRIGGER IF EXISTS trg\_store\_deleted\_show ON "Show";

CREATE TRIGGER trg\_store\_deleted\_show

BEFORE DELETE ON "Show"

FOR EACH ROW

EXECUTE FUNCTION store\_deleted\_show('deleted\_by\_app');

-- 6. TRIGGER: keep Show.total\_duration updated when Episode inserted or deleted

-- Helper functions

CREATE OR REPLACE FUNCTION \_add\_episode\_duration()

RETURNS TRIGGER LANGUAGE plpgsql AS $$

BEGIN

UPDATE "Show" SET total\_duration = COALESCE(total\_duration,'00:00:00') + NEW.duration

WHERE show\_id = NEW.show\_id;

RETURN NEW;

END;

$$;

CREATE OR REPLACE FUNCTION \_remove\_episode\_duration()

RETURNS TRIGGER LANGUAGE plpgsql AS $$

BEGIN

UPDATE "Show" SET total\_duration = COALESCE(total\_duration,'00:00:00') - OLD.duration

WHERE show\_id = OLD.show\_id;

RETURN OLD;

END;

$$;

DROP TRIGGER IF EXISTS trg\_episode\_insert ON "Episode";

CREATE TRIGGER trg\_episode\_insert

AFTER INSERT ON "Episode"

FOR EACH ROW

EXECUTE FUNCTION \_add\_episode\_duration();

DROP TRIGGER IF EXISTS trg\_episode\_delete ON "Episode";

CREATE TRIGGER trg\_episode\_delete

AFTER DELETE ON "Episode"

FOR EACH ROW

EXECUTE FUNCTION \_remove\_episode\_duration();

-- 7. TRIGGER: Prevent TRP from exceeding 10 (on Marketing/trp insert/update)

CREATE OR REPLACE FUNCTION prevent\_trp\_over\_10()

RETURNS TRIGGER LANGUAGE plpgsql AS $$

BEGIN

IF NEW.trp IS NOT NULL AND NEW.trp > 10 THEN

RAISE EXCEPTION 'TRP cannot exceed 10. Attempted: %', NEW.trp;

END IF;

RETURN NEW;

END;

$$;

DROP TRIGGER IF EXISTS trg\_marketing\_trp ON "Marketing";

CREATE TRIGGER trg\_marketing\_trp

BEFORE INSERT OR UPDATE ON "Marketing"

FOR EACH ROW

EXECUTE FUNCTION prevent\_trp\_over\_10();

-- 8. PROCEDURE: promote upcoming show to Show (auto-validate fields)

CREATE OR REPLACE PROCEDURE promote\_upcoming\_show(p\_upcoming\_id INT)

LANGUAGE plpgsql

AS $$

DECLARE

v\_title VARCHAR;

v\_channel INT;

v\_director INT;

v\_replace INT;

BEGIN

SELECT title, channel\_id, director\_id, replacement\_show\_id INTO v\_title, v\_channel, v\_director, v\_replace

FROM "UpcomingShow" WHERE upcoming\_id = p\_upcoming\_id;

IF v\_title IS NULL THEN

RAISE EXCEPTION 'Upcoming show not found';

END IF;

-- Insert into Show

INSERT INTO "Show"(channel\_id, genre\_id, title, air\_time, director\_id)

VALUES (v\_channel, NULL, v\_title, NULL, v\_director)

RETURNING show\_id INTO v\_replace;

-- Optionally update replacement\_show\_id in UpcomingShow and delete upcoming entry:

DELETE FROM "UpcomingShow" WHERE upcoming\_id = p\_upcoming\_id;

END;

$$;

-- 9. PROCEDURE: add show with validation (example: ensure channel exists, title unique in channel)

CREATE OR REPLACE PROCEDURE add\_show\_valid(

p\_channel\_id INT,

p\_genre\_id INT,

p\_title VARCHAR,

p\_air\_time TIME,

p\_director\_id INT)

LANGUAGE plpgsql

AS $$

DECLARE

cnt INT;

BEGIN

-- check channel exists

SELECT COUNT(\*) INTO cnt FROM "Channel" WHERE channel\_id = p\_channel\_id;

IF cnt = 0 THEN

RAISE EXCEPTION 'Channel % does not exist', p\_channel\_id;

END IF;

-- check duplicate title on same channel

SELECT COUNT(\*) INTO cnt FROM "Show" WHERE channel\_id = p\_channel\_id AND title = p\_title;

IF cnt > 0 THEN

RAISE EXCEPTION 'Show % already exists on channel %', p\_title, p\_channel\_id;

END IF;

INSERT INTO "Show"(channel\_id, genre\_id, title, air\_time, director\_id)

VALUES (p\_channel\_id, p\_genre\_id, p\_title, p\_air\_time, p\_director\_id);

END;

$$;

-- 10. FUNCTION: average episode duration per season for a show (returns set)

CREATE OR REPLACE FUNCTION avg\_episode\_duration\_per\_season(p\_show\_id INT)

RETURNS TABLE(season\_number INT, avg\_duration INTERVAL)

LANGUAGE SQL

AS $$

SELECT season\_number, AVG(duration) AS avg\_duration

FROM "Episode"

WHERE show\_id = p\_show\_id

GROUP BY season\_number

ORDER BY season\_number;

$$;

-- 11. QUERY helper: shows with trp above average (we'll run as query later)

-- 12. Directors who directed multiple genres (query later)

-- 13. Recursive query for shows where director is also actor (example query later)

-- 12b. Create a schedule table (for procedure generating weekly schedule)

CREATE TABLE IF NOT EXISTS ShowSchedule (

schedule\_id SERIAL PRIMARY KEY,

show\_id INT REFERENCES "Show"(show\_id) ON DELETE CASCADE,

channel\_id INT REFERENCES "Channel"(channel\_id) ON DELETE CASCADE,

scheduled\_date DATE,

scheduled\_time TIME,

created\_at TIMESTAMP DEFAULT now()

);

-- 13. PROCEDURE: generate weekly schedule for a channel (very simple round-robin of shows)

CREATE OR REPLACE PROCEDURE generate\_weekly\_schedule(p\_channel\_id INT, p\_start\_date DATE)

LANGUAGE plpgsql

AS $$

DECLARE

rec RECORD;

idx INT := 0;

shows CURSOR FOR SELECT show\_id FROM "Show" WHERE channel\_id = p\_channel\_id ORDER BY show\_id;

cur\_date DATE := p\_start\_date;

BEGIN

OPEN shows;

LOOP

FETCH shows INTO rec;

EXIT WHEN NOT FOUND;

idx := idx + 1;

INSERT INTO ShowSchedule(show\_id, channel\_id, scheduled\_date, scheduled\_time)

VALUES (rec.show\_id, p\_channel\_id, cur\_date + (idx / 24)::int, (INTERVAL '1 hour' \* (idx % 24))::time);

-- naive: schedule spaced by 1 hour for demo purposes

IF idx % 24 = 0 THEN

cur\_date := cur\_date + 1;

END IF;

END LOOP;

CLOSE shows;

END;

$$;

-- 14. TRIGGER: on Marketing insert/update, ensure TRP valid (already created), also

-- create trigger to prevent negative cost

CREATE OR REPLACE FUNCTION prevent\_negative\_cost()

RETURNS TRIGGER LANGUAGE plpgsql AS $$

BEGIN

IF NEW.cost\_of\_production < 0 THEN

RAISE EXCEPTION 'Cost cannot be negative';

END IF;

RETURN NEW;

END;

$$;

DROP TRIGGER IF EXISTS trg\_marketing\_cost ON "Marketing";

CREATE TRIGGER trg\_marketing\_cost

BEFORE INSERT OR UPDATE ON "Marketing"

FOR EACH ROW

EXECUTE FUNCTION prevent\_negative\_cost();

-- 15. SAMPLE VIEW "show\_details" to simplify some queries

CREATE OR REPLACE VIEW show\_details AS

SELECT s.show\_id, s.title, s.channel\_id, c.channel\_name, s.genre\_id, g.genre\_name, s.total\_duration

FROM "Show" s

LEFT JOIN "Channel" c ON s.channel\_id = c.channel\_id

LEFT JOIN "Genre" g ON s.genre\_id = g.genre\_id;

QUERY RESULT:

<!doctype html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Query Result</title>

  <style>

    body {

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      background: url('q.jpg') no-repeat center center fixed;

      background-size: cover;

      color: #fff;

      margin: 0;

      padding: 0;

    }

    .overlay {

      background-color: rgba(0, 0, 0, 0.6);

      min-height: 100vh;

      padding: 40px;

      text-align: center;

    }

    h2 {

      font-size: 2.5em;

      margin-bottom: 20px;

      color: #ffcc70;

      text-shadow: 0 0 10px #000;

    }

    table {

      width: 80%;

      margin: 30px auto;

      border-collapse: collapse;

      background: rgba(255, 255, 255, 0.1);

      border-radius: 12px;

      overflow: hidden;

      box-shadow: 0 4px 30px rgba(0, 0, 0, 0.4);

    }

    th, td {

      padding: 12px 20px;

      border-bottom: 1px solid rgba(255, 255, 255, 0.2);

      color: #fff;

    }

    tr:hover {

      background: rgba(255, 255, 255, 0.2);

      transition: 0.3s ease;

    }

    a {

      display: inline-block;

      margin-top: 30px;

      padding: 10px 25px;

      background: #ffcc70;

      color: #000;

      text-decoration: none;

      font-weight: bold;

      border-radius: 25px;

      transition: 0.3s;

    }

    a:hover {

      background: #fff;

      color: #333;

      box-shadow: 0 0 10px #ffcc70;

    }

    @media (max-width: 768px) {

      table {

        width: 95%;

        font-size: 14px;

      }

      h2 {

        font-size: 1.8em;

      }

    }

  </style>

</head>

<body>

  <div class="overlay">

    <h2>Query: {{ name }}</h2>

    <table>

      {% for r in rows %}

      <tr>

        {% for c in r %}

        <td>{{ c }}</td>

        {% endfor %}

      </tr>

      {% endfor %}

    </table>

    <a href="/queries">Back</a>

  </div>

</body>

</html>