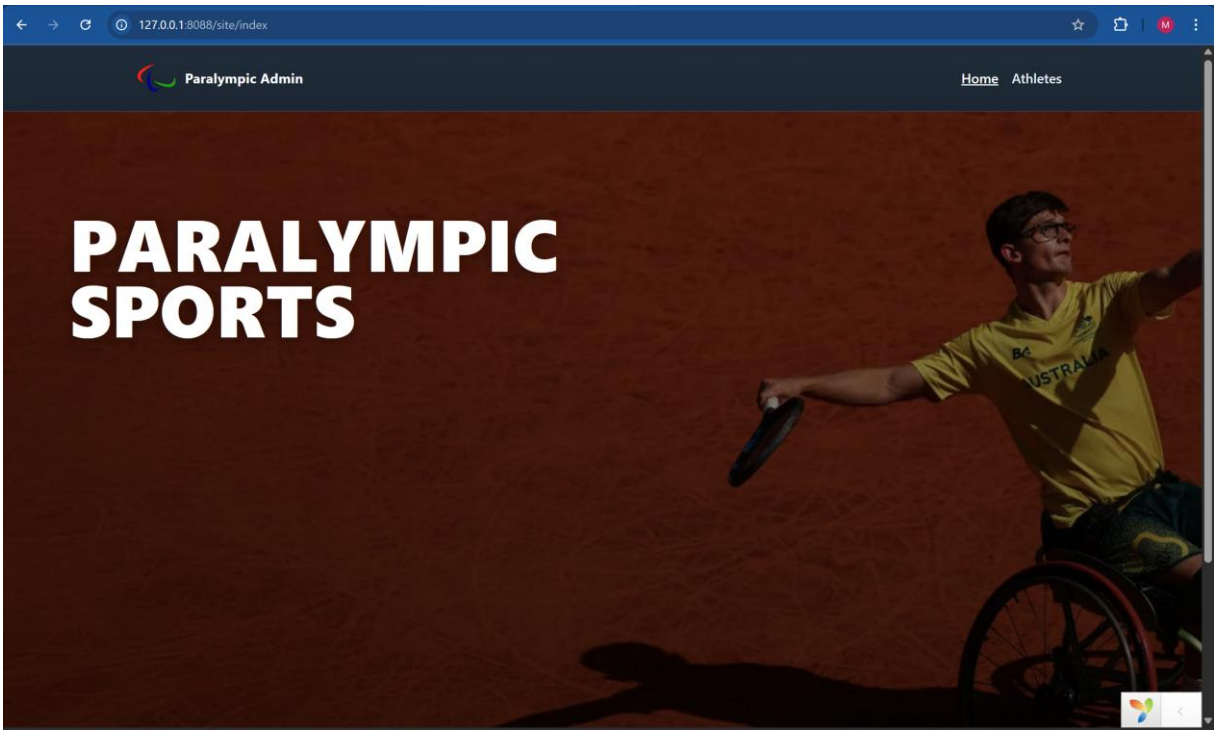
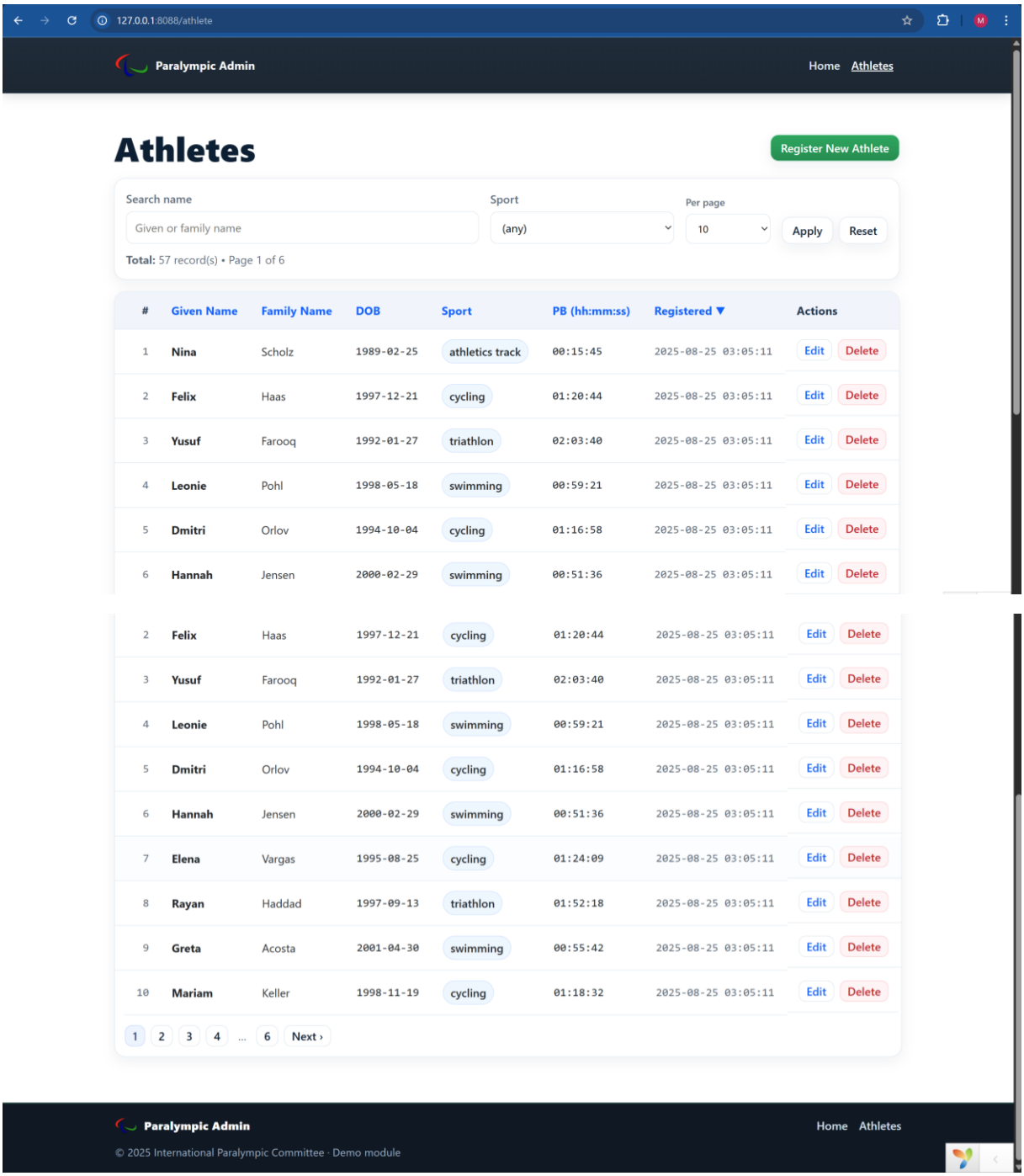
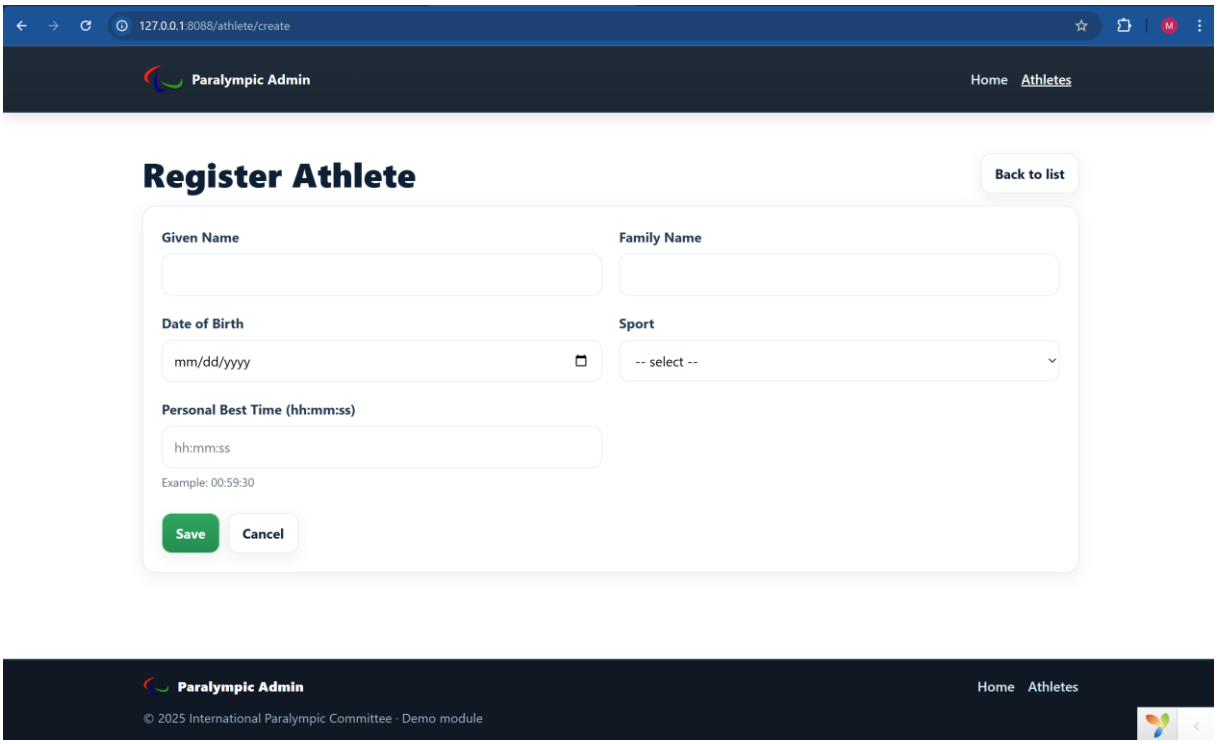
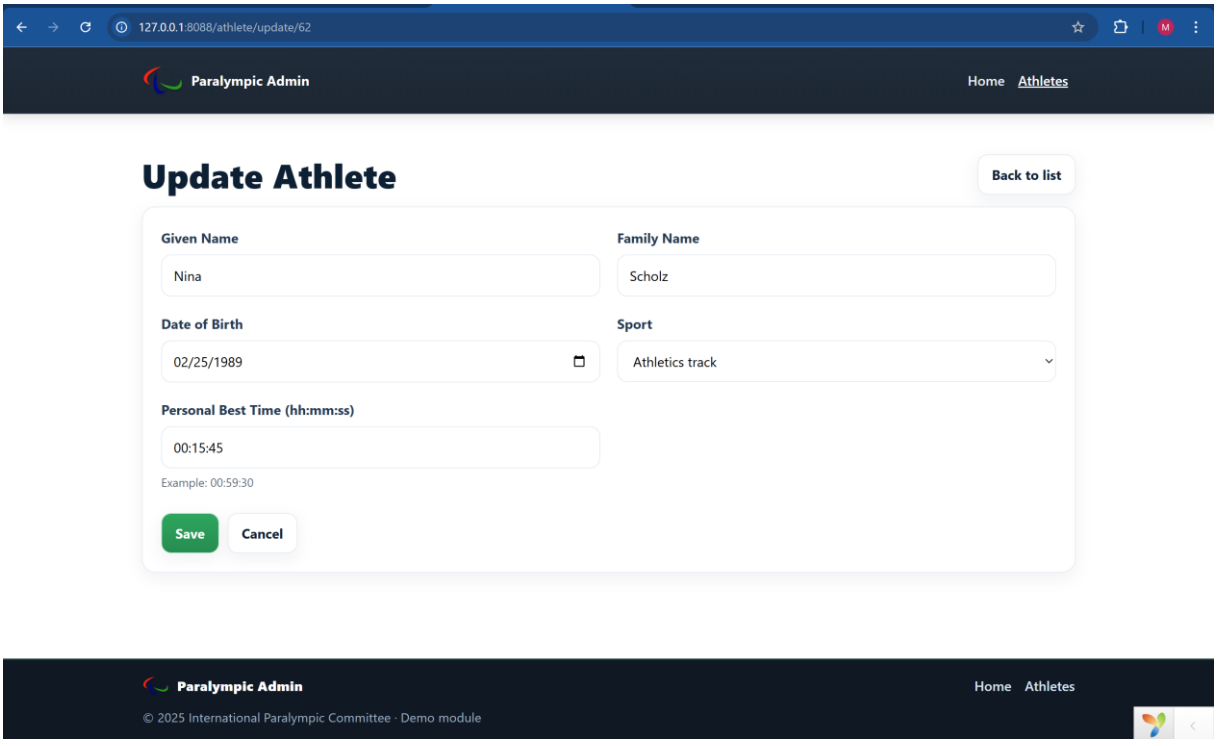
Task 1

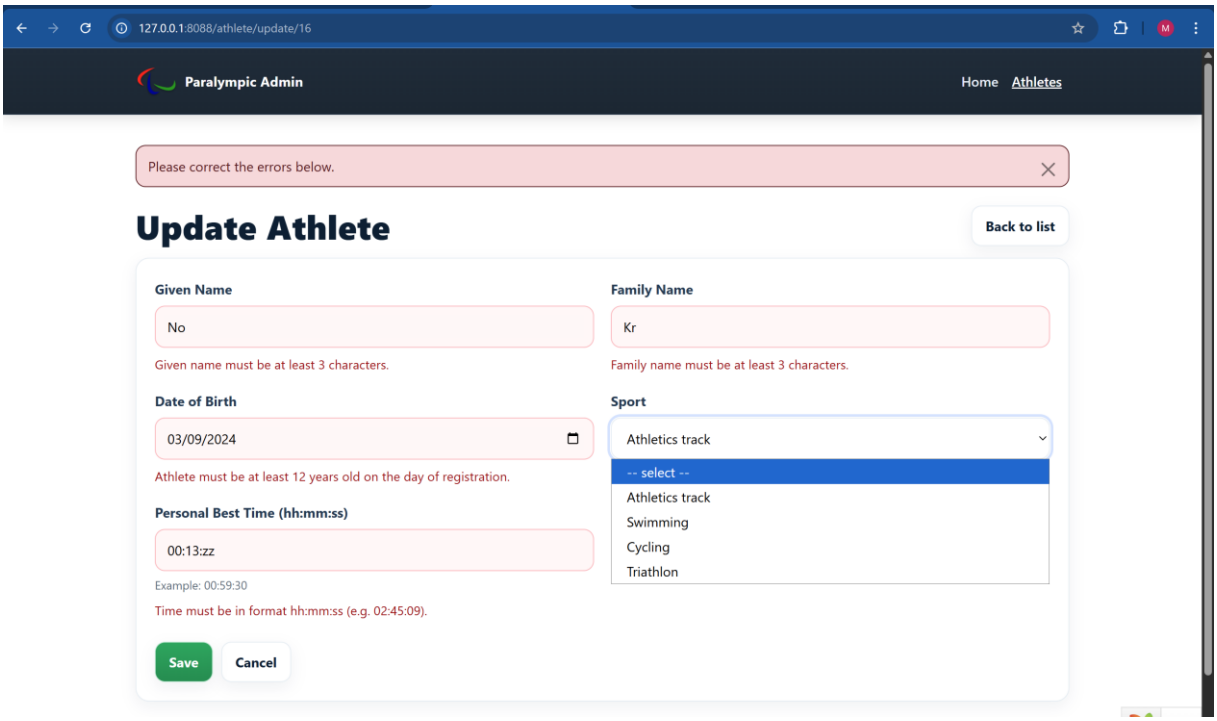
Home Page:

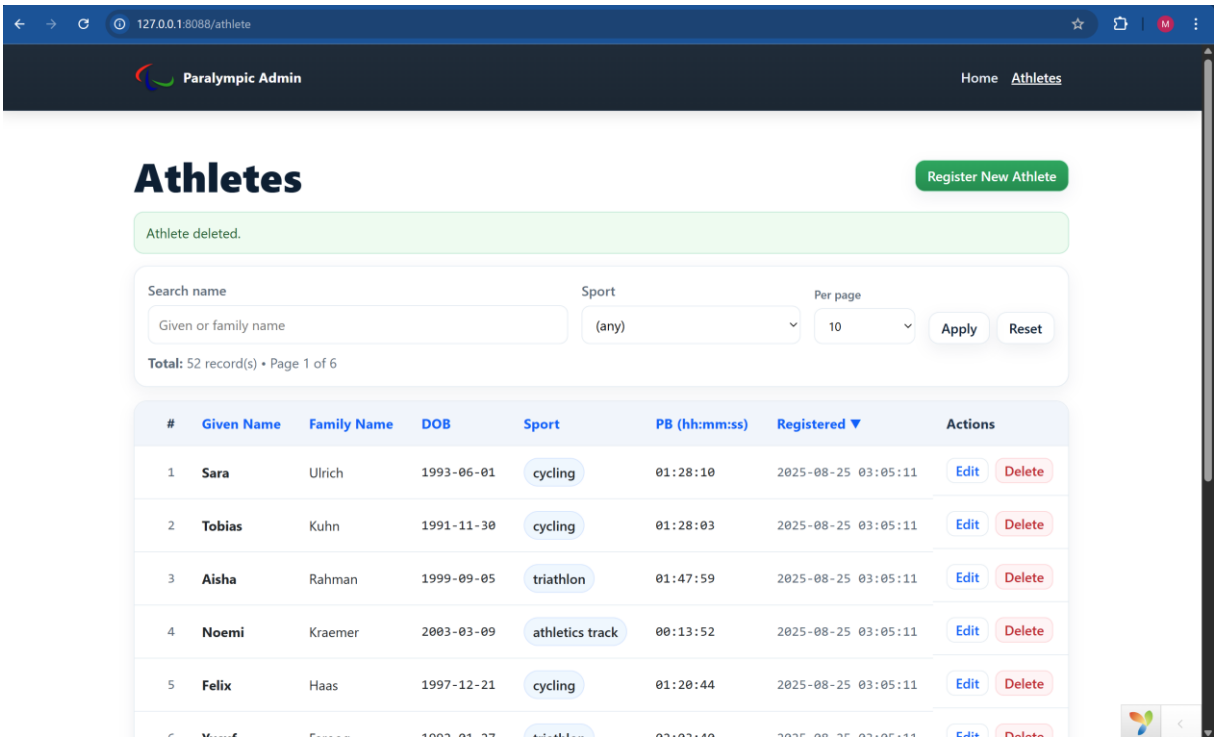



127.0.0.1:8088/athlete
Paralympic Admin
Home Athletes
Athletes
Register New Athlete
Search name
Given or family name
Sport
(any)
Per page
10
Apply
Reset
Total: 57 record(s) • Page 1 of 6
Given Name Family Name DOB Sport PB (hh:mm:ss) Registered ▼ Actions
1 Nina Scholz 1989-02-25 athletics track 00:15:45 2025-08-25 03:05:11 Edit Delete
2 Felix Haas 1997-12-21 cycling 01:20:44 2025-08-25 03:05:11 Edit Delete
3 Yusuf Farooq 1992-01-27 triathlon 02:03:40 2025-08-25 03:05:11 Edit Delete
4 Leonie Pohl 1998-05-18 swimming 00:59:21 2025-08-25 03:05:11 Edit Delete
5 Dmitri Orlov 1994-10-04 cycling 01:16:58 2025-08-25 03:05:11 Edit Delete
6 Hannah Jensen 2000-02-29 swimming 00:51:36 2025-08-25 03:05:11 Edit Delete
7 Elena Vargas 1995-08-25 cycling 01:24:09 2025-08-25 03:05:11 Edit Delete
8 Rayan Haddad 1997-09-13 triathlon 01:52:18 2025-08-25 03:05:11 Edit Delete
9 Greta Acosta 2001-04-30 swimming 00:55:42 2025-08-25 03:05:11 Edit Delete
10 Mariam Keller 1998-11-19 cycling 01:18:32 2025-08-25 03:05:11 Edit Delete
1 2 3 4 ... 6 Next ›
Paralympic Admin
Home Athletes
© 2025 International Paralympic Committee · Demo module


127.0.0.1:8088/athlete/create
Paralympic Admin
Home
Athletes
Register Athlete
Back to list
Given Name
Family Name
Date of Birth
mm/dd/yyyy
Sport
-- select --
Personal Best Time (hh:mm:ss)
hh:mm:ss
Example: 00:59:30
Save
Cancel
Paralympic Admin
Home
Athletes
© 2025 International Paralympic Committee · Demo module


127.0.0.1:8088/athlete/update/62
Paralympic Admin
Home
Athletes
Update Athlete
Back to list
Given Name
Nina
Family Name
Scholz
Date of Birth
02/25/1989
Sport
Athletics track
Personal Best Time (hh:mm:ss)
00:15:45
Example: 00:59:30
Save
Cancel
Paralympic Admin
Home
Athletes
© 2025 International Paralympic Committee · Demo module


127.0.0.1:8088/athlete/update/16
Paralympic Admin
Home
Athletes
Please correct the errors below.
Update Athlete
Back to list
Given Name
No
Given name must be at least 3 characters.
Family Name
Kr
Family name must be at least 3 characters.
Date of Birth
03/09/2024
Athlete must be at least 12 years old on the day of registration.
Sport
Athletics track
-- select --
Athletics track
Swimming
Cycling
Triathlon
Personal Best Time (hh:mm:ss)
00:13:zz
Example: 00:59:30
Time must be in format hh:mm:ss (e.g. 02:45:09).
Save
Cancel


127.0.0.1:8088/athlete
Paralympic Admin
Home
Athletes
Athletes
Register New Athlete
Athlete deleted.
Search name
Given or family name
Sport
(any)
Per page
10
Apply
Reset
Total: 52 record(s) • Page 1 of 6
Given Name Family Name DOB Sport PB (hh:mm:ss) Registered ▼ Actions
1 Sara Ulrich 1993-06-01 cycling 01:28:10 2025-08-25 03:05:11 Edit Delete
2 Tobias Kuhn 1991-11-30 cycling 01:28:03 2025-08-25 03:05:11 Edit Delete
3 Aisha Rahman 1999-09-05 triathlon 01:47:59 2025-08-25 03:05:11 Edit Delete
4 Noemi Kraemer 2003-03-09 athletics track 00:13:52 2025-08-25 03:05:11 Edit Delete
5 Felix Haas 1997-12-21 cycling 01:20:44 2025-08-25 03:05:11 Edit Delete

TASK 2

Part A

ERD DIAGRAM LINK

TASK 2

Part B

1) Who are the top eight athletes of the Athletics Men's 1500m Semi-final 2 race?

```
SELECT a.athlete_id,
      a.family_name, a.given_name,
      n.code AS noc,
      p.rank, p.time_ms
FROM Sport s
JOIN EventTemplate et   ON et.sport_id = s.sport_id
JOIN EventEdition ee    ON ee.eventtemplate_id = et.eventtemplate_id
JOIN Round r            ON r.eventedition_id = ee.eventedition_id
JOIN CompetitionUnit cu ON cu.round_id = r.round_id
JOIN Participation p    ON p.competitionunit_id = cu.competitionunit_id
JOIN AthleteCompetitor ac ON ac.competitor_id = p.competitor_id
JOIN Athlete a          ON a.athlete_id = ac.athlete_id
JOIN NOC n              ON n.nocid = a.nocid
WHERE ee.gamesid = :games
  AND s.name = 'Athletics'
  AND et.name = '1500m'
  AND et.gender = 'Men'
  AND r.name = 'Semi-final 2'
ORDER BY p.rank, p.time_ms
FETCH FIRST 8 ROWS ONLY;
```

Explanation:

I start on **Sport** and filter to *Athletics*, then follow **EventTemplate → EventEdition** for the chosen Games. From the **Round** labeled *Semifinal 2* I jump to its **CompetitionUnit**. Placings live in **Participation**, so I join that to **Competitor → AthleteCompetitor → Athlete**, and add **NOC** for the flag. I sort by Participation.rank (breaking ties with time when present) and take the first eight. This walks the center spine (Template→Edition→Round→Unit) and uses **Participation** exactly as I modeled it.

2) Who were the opponents the gold medallist of the Table Tennis Women's Individual had to face during the tournament, and what were the final scores?

```
WITH medalist AS (
  SELECT ee.eventedition_id, ma.competitor_id
  FROM Sport s
  JOIN EventTemplate et ON et.sport_id = s.sport_id
```

```
  JOIN EventEdition ee  ON ee.eventtemplate_id = et.eventtemplate_id
  JOIN MedalAwards ma   ON ma.eventedition_id = ee.eventedition_id
  WHERE s.name='Table Tennis'
    AND et.name='Women"s Singles'
    AND ma.medal='Gold'
    AND ee.gamesid=:games
)
SELECT cu.competitionunit_id,
       r.name AS round_name,
       COALESCE(a2.family_name || ' ' || a2.given_name, t2.name) AS opponent,
       STRING_AGG(ss.points_for::text || '-' || ss.points_against::text
                  ORDER BY ss.set_no) AS scoreline
FROM medalist m
JOIN Round r           ON r.eventedition_id = m.eventedition_id
JOIN CompetitionUnit cu ON cu.round_id = r.round_id
JOIN Participation p1   ON p1.competitionunit_id = cu.competitionunit_id
                       AND p1.competitor_id = m.competitor_id
JOIN Participation p2   ON p2.competitionunit_id = cu.competitionunit_id
                       AND p2.competitor_id <> p1.competitor_id
LEFT JOIN AthleteCompetitor ac2 ON ac2.competitor_id = p2.competitor_id
LEFT JOIN Athlete a2           ON a2.athlete_id = ac2.athlete_id
LEFT JOIN TeamCompetitor tc2    ON tc2.competitor_id = p2.competitor_id
LEFT JOIN Team t2              ON t2.team_id = tc2.team_id
JOIN SetScore ss ON ss.competitionunit_id = cu.competitionunit_id
                AND ss.competitor_id     = p1.competitor_id
GROUP BY cu.competitionunit_id, r.name, opponent
ORDER BY MIN(r.seq_no);
```

Explanation:

I find the champion through **MedalAwards** attached to the event's **EventEdition** where **EventTemplate** is *Table Tennis / Women / Singles* and medal='Gold'. That gives me her competitor_id (an **AthleteCompetitor**). For each match she played, I fetch all **CompetitionUnit** rows where she appears in **Participation**, then self-join **Participation** on the same unit to get the opponent. I pull set-by-set scores from **SetScore** and aggregate to a match scoreline. This shows why I separated **SetScore** under **Participation**—racquet sports need it while track events don't.

3) In which football matches did the second goalkeeper from Great Britain compete?

```
WITH ranked_gk AS (
  SELECT tm.*,
         ROW_NUMBER() OVER (PARTITION BY tm.team_id, tm.gamesid
                            ORDER BY tm.jersey_number NULLS LAST) AS rn
  FROM TeamMembership tm
  JOIN Athlete a ON a.athlete_id = tm.athlete_id
  JOIN NOC n     ON n.nocid = a.nocid
  WHERE n.code='GBR' AND tm.role='GK' AND tm.gamesid=:games
```

```
)
SELECT cu.competitionunit_id, r.name AS round_name, p.minutes_played, p.started
FROM ranked_gk gk
JOIN TeamCompetitor tc ON tc.team_id = gk.team_id
JOIN Participation p     ON p.competitor_id = tc.competitor_id
JOIN CompetitionUnit cu ON cu.competitionunit_id = p.competitionunit_id
JOIN Round r             ON r.round_id = cu.round_id
JOIN EventEdition ee     ON ee.eventedition_id = r.eventedition_id
JOIN EventTemplate et    ON et.eventtemplate_id = ee.eventtemplate_id
JOIN Sport s             ON s.sport_id = et.sport_id
WHERE gk.rn = 2 AND s.name = 'Football' AND ee.gamesid=:games
ORDER BY cu.competitionunit_id;
```

Explanation:

Squad roles live in **TeamMembership**, so I filter NOC='GBR' and role='GK', then use a window function to select the *second* keeper on that team for the Games. From that athlete I resolve to **Competitor/TeamCompetitor** and list every **CompetitionUnit** they actually appeared in via **Participation**. I reattach **Round/Edition/Template** and keep Sport='Football' to ensure I'm only listing football matches. This leverages **TeamMembership** for roster context and **Participation** for minutes on the pitch.

4) Who are the female silver medallists who have competed in gender mixed events?

```
SELECT DISTINCT a.athlete_id, a.family_name, a.given_name, n.code AS noc
FROM MedalRecipient mr
JOIN MedalAwards ma ON ma.eventedition_id = mr.eventedition_id
                   AND ma.competitor_id  = mr.competitor_id
JOIN Athlete a  ON a.athlete_id = mr.athlete_id
JOIN NOC n      ON n.nocid = a.nocid
JOIN EventEdition ee ON ee.eventedition_id = ma.eventedition_id
WHERE ma.medal = 'Silver' AND a.sex = 'F' AND ee.gamesid = :games
  AND EXISTS (
     SELECT 1
     FROM Registration r2
     JOIN EventEdition ee2  ON ee2.eventedition_id = r2.eventedition_id
     JOIN EventTemplate et2 ON et2.eventtemplate_id = ee2.eventtemplate_id
     JOIN Participation p2  ON p2.competitor_id = r2.competitor_id
     JOIN AthleteCompetitor ac2 ON ac2.competitor_id = r2.competitor_id
     WHERE ac2.athlete_id = a.athlete_id
       AND et2.gender = 'Mixed'
       AND ee2.gamesid = ee.gamesid
  )
ORDER BY n.code, a.family_name, a.given_name;
```

Explanation:

I read **MedalAwards** joined through **MedalRecipient** to the credited **Competitor**, then down **ISA** to **Athlete**. I keep sex='F' and add **NOC**. To prove a mixed-event appearance, I use an EXISTS against **Registration** (or **Participation**) on any **EventEdition** in the same Games where **EventTemplate.gender='Mixed'`. This cleanly reuses the "who got the medal" path (**MedalRecipient**) and the "who entered/played" path (**Registration/Participation**).

5) How many medals have been awarded to African boxers?

```
SELECT COUNT(*) AS medals_to_african_boxers
FROM MedalAwards ma
JOIN EventEdition ee  ON ee.eventedition_id = ma.eventedition_id
JOIN EventTemplate et ON et.eventtemplate_id = ee.eventtemplate_id
JOIN Sport s          ON s.sport_id = et.sport_id
JOIN AthleteCompetitor ac ON ac.competitor_id = ma.competitor_id
JOIN Athlete a        ON a.athlete_id = ac.athlete_id
JOIN NOC n            ON n.nocid = a.nocid
WHERE s.name='Boxing' AND n.continent='Africa' AND ee.gamesid=:games;
```

Explanation:

I limit **MedalAwards** to **EventTemplate.Sport='Boxing'**, then credit via **MedalRecipient** to the real recipient competitor. Through **ISA** I land on **AthleteCompetitor → Athlete**, attach **NOC**, and filter to African NOCs (using a continent/region attribute). I group by medal color for a tidy medal table. The same pattern works for any sport/region without changing the schema.

6) How many athletes competed for India at these Olympic Games?

```
WITH ind_ath AS (
  -- Individual entries
  SELECT DISTINCT ac.athlete_id
  FROM AthleteCompetitor ac
  JOIN Athlete a ON a.athlete_id = ac.athlete_id
  JOIN NOC n     ON n.nocid = a.nocid AND n.code='IND'
  JOIN Participation p    ON p.competitor_id = ac.competitor_id
  JOIN CompetitionUnit cu ON cu.competitionunit_id = p.competitionunit_id
  JOIN Round r            ON r.round_id = cu.round_id
  JOIN EventEdition ee    ON ee.eventedition_id = r.eventedition_id
  WHERE ee.gamesid=:games
  UNION
  -- Via teams
  SELECT DISTINCT tm.athlete_id
  FROM TeamMembership tm
  JOIN Team t         ON t.team_id = tm.team_id
  JOIN NOC n          ON n.nocid = t.nocid AND n.code='IND'
  JOIN TeamCompetitor tc ON tc.team_id = t.team_id
  JOIN Participation p   ON p.competitor_id = tc.competitor_id
```

```
  JOIN CompetitionUnit cu ON cu.competitionunit_id = p.competitionunit_id
  JOIN Round r           ON r.round_id = cu.round_id
  JOIN EventEdition ee    ON ee.eventedition_id = r.eventedition_id
  WHERE ee.gamesid=:games AND tm.gamesid=:games
)
SELECT COUNT(*) AS athletes_for_IND
FROM ind_ath;
```

Explanation:

To count *competed* (not just registered), I require at least one **Participation** in a **CompetitionUnit** for the Games. I join **Participation → Competitor → AthleteCompetitor → Athlete**, attach **NOC**, filter NOC='IND', and COUNT(DISTINCT athlete_id). Using Participation prevents counting alternates or withdrawals.

7) How old was the youngest athletes competed in any of the gymnastics finals?

```
SELECT a.athlete_id, a.family_name, a.given_name, n.code AS noc,
      MIN(DATE_PART('year', AGE(cu.start_time, a.dob))) AS age_years
FROM Sport s
JOIN EventTemplate et   ON et.sport_id = s.sport_id
JOIN EventEdition ee    ON ee.eventtemplate_id = et.eventtemplate_id
JOIN Round r           ON r.eventedition_id = ee.eventedition_id
JOIN CompetitionUnit cu ON cu.round_id = r.round_id
JOIN Participation p    ON p.competitionunit_id = cu.competitionunit_id
JOIN AthleteCompetitor ac ON ac.competitor_id = p.competitor_id
JOIN Athlete a          ON a.athlete_id = ac.athlete_id
JOIN NOC n              ON n.nocid = a.nocid
WHERE s.name LIKE 'Gymnastics%'
  AND ee.gamesid = :games
  AND r.name ILIKE '%Final%'
GROUP BY a.athlete_id, a.family_name, a.given_name, n.code
ORDER BY age_years ASC
FETCH FIRST 1 ROW ONLY;
```

Explanation:

I filter **Sport='Gymnastics'** and follow **EventTemplate → EventEdition → Round** where round_name='Final'. From the corresponding **CompetitionUnit** rows I collect athletes via **Participation → Competitor → AthleteCompetitor → Athlete**. I compute age at competition from CompetitionUnit.scheduled_start - Athlete.birth_date and take the minimum, reporting the athlete, event, and NOC. Storing the unit's timestamp is what lets me answer "age on the day" precisely.

Part C

I wouldn't redesign what's already working. The competition spine (Sport → Discipline → EventTemplate → EventEdition → Round → CompetitionUnit) and the "who competes" stack (NOC → Athlete/Team → Competitor with ISA to AthleteCompetitor/TeamCompetitor) stay exactly as they are. What I add is a small, self-contained **classification layer** that plugs into both sides and enforces the three Paralympic rules at the point where they matter: when someone tries to enter or compete.

- **Each Para athlete is "classified" in up to three different "classes".**

I introduce a **Class** entity (e.g., T12, F46) that is scoped to **Discipline**. That choice is deliberate: classification schemes are discipline-specific, so tying Class to Discipline prevents collisions between, say, track ("T") and field ("F") codes and lets me version descriptions as the IPC updates guidance. Between **Athlete** and **Class** I add **AthleteClassification** (bridge table) with attributes such as status (Confirmed/Review), assessed_on, valid_until, and an optional primary_flag. I do **not** denormalize the athlete table with class1/class2/class3 columns; the "up to three" is a business rule, not structure, so I enforce it with a simple cap (trigger or constraint) that rejects a fourth row for the same athlete. This keeps history tidy—multiple assessments over time—without duplicating data.

- **Each medal event defines one or multiple eligible classes. An athlete must have at least one of those classes to be eligible to compete in that medal event.**

Eligibility changes between Games, so I anchor it at **EventEdition** (the specific Paris 2024 or LA 2028 instance), not at EventTemplate. I add **EligibleClass** (bridge: EventEdition ↔ Class). That gives each edition its own eligibility list and preserves history if rules evolve. Practically, it creates a very short, readable join path from any event instance to the set of class codes permitted on its start list and medal table.

- **For your better understanding of athlete's and event's classes, …….**

I don't scatter flags across the schema; I enforce this rule where participation becomes real—**Registration** and **Participation**. When a registration is attempted, I require an EXISTS overlap between that athlete's classifications and the event's eligible classes (and that the classification is active and in a valid status). If there's no overlap, the insert fails with a clear error. This approach keeps the core model simple and prevents bad data *before* it lands in start lists or results. For team events, I apply the same logic to the team's **TeamMembership** for that Games: every athlete named to play must meet the event's eligibility list; if a sport later needs "points-on-court" constraints, I can add a small optional constraint table per EventTemplate and evaluate it in the same gate.

This design is strong because its minimal (one domain table, two bridges, one gate), so it doesn't disturb medals, rounds, scores, or any queries I've already written for the Olympics. It's accurate: discipline-scoped classes mirror how the IPC actually publishes lists, and edition-scoped eligibility matches how programs differ by Games. It's auditable: because eligibility lives at EventEdition, I can always reconstruct "who was eligible" for a past Games. It's performant: I index (athlete_id, class_id) and (event_edition_id, class_id) so the

eligibility check is a tiny intersection, even at scale. And it's extensible: if a federation adds new codes or merges classifications, I update the Class table with versions—no schema churn.

On the ERD, I place **Class** just left of **Athlete**. I add a diamond "HAS_CLASS" (Athlete ↔ Class) with the bridge box **AthleteClassification** beneath it and a small note "≤ 3 per athlete (business rule)". From the center stack, I add a diamond "ELIGIBLE_CLASS" between **EventEdition** and **Class**, with the bridge box **EligibleClass**. Everything else remains as-is. That visual communicates two things at a glance: one is classification belongs to people, and 2nd is eligibility belongs to specific event editions.

I can publish eligible entrant lists by joining EventEdition → EligibleClass → AthleteClassification; I can validate entries in real time; and I can report medals by class with one extra join. Most importantly, I don't fork my data model for Paralympics—I **extend** it—so existing Olympic queries, APIs, and ETL continue to work with a single additional join when the context is Para. That's the kind of change that's easy to explain, easy to maintain, and hard to break.