## 🔹 #1

### 1️⃣ Первый запрос

### 🔹 1. Общая цель запроса

Цель — определить **средний возраст спортсменов** для каждого **вида медали** (золото, серебро, бронза), исключая записи, где медаль отсутствует или помечена как «NA».

### 🔹 2. Источники данных

В запросе участвуют три таблицы из схемы olympics:

1. **medal** — содержит сведения о типах медалей (например, название медали и её идентификатор).
2. **games\_competitor** — информация об участниках определённых Олимпийских игр (включая возраст и уникальный идентификатор участника).
3. **competitor\_event** — связывает участников с конкретными событиями, а также указывает, получили ли они медаль и какую именно.

### 🔹 3. Внутренние соединения (INNER JOIN)

Происходит двойное соединение:

1. **Первое соединение** — между games\_competitor и competitor\_event:  
   таким образом сопоставляются участники и события, в которых они участвовали.
2. **Второе соединение** — между результатом этого объединения и таблицей medal:  
   используется поле medal\_id, чтобы к каждому событию добавить информацию о типе медали, полученной участником.

Благодаря этим соединениям формируется единая выборка, где каждая строка отражает:

* конкретного участника,
* событие,
* и тип полученной медали.

### 🔹 4. Условие отбора (WHERE)

Добавлено ограничение, исключающее строки, где значение medal\_name равно 'NA'.  
Это делается для того, чтобы не учитывать случаи отсутствия награды — в анализ включаются только реальные медали.

### 🔹 5. Агрегация данных

После объединения и фильтрации выполняется группировка (GROUP BY) по названию медали (medal\_name).  
Это означает, что все участники, получившие одну и ту же медаль, объединяются в одну аналитическую категорию.

### 🔹 6. Вычисление среднего возраста

Для каждой группы (то есть для каждого типа медали) вычисляется средний возраст участников с помощью функции AVG().  
Результат округляется функцией ROUND() для удобства представления — чтобы средние значения выглядели аккуратно и не содержали избыточных десятичных знаков.

### 🔹 7. Результат выполнения

На выходе формируется таблица, где каждая строка соответствует определённому типу медали, а напротив указано среднее значение возраста спортсменов, получивших эту медаль.

Таким образом, запрос даёт сводную статистику по возрастному составу призёров Олимпийских игр в разрезе категорий наград.

### 2️⃣ Второй запрос

**Общая цель:**  
Определить 5 регионов с наибольшим количеством уникальных участников, участвовавших более чем в трёх разных соревнованиях.

**Логика:**

1. **Основная таблица:**  
   noc\_region — список регионов (Национальные олимпийские комитеты).
2. **Соединение:**  
   С person\_region, чтобы связать регион с конкретными людьми (участниками).
3. **Подзапрос в WHERE:**  
   Отбирает только тех участников (competitor\_id), которые участвовали более чем в 3 различных соревнованиях (event\_id), используя HAVING COUNT(DISTINCT ce.event\_id) > 3.
4. **Подсчёт:**  
   COUNT(DISTINCT pr.person\_id) определяет количество уникальных участников в каждом регионе, соответствующих условию.
5. **Сортировка и ограничение:**  
   ORDER BY unique\_competitors DESC — регионы с наибольшим числом таких участников.  
   LIMIT 5 — только пять лучших регионов.

### 3️⃣ Третий блок

**Создание временной таблицы temp\_medal\_count.**

**Цель:**  
Посчитать, сколько медалей получил каждый участник.

**Действия:**

1. Из таблицы competitor\_event выбираются записи, где medal\_id не равен NULL, то есть есть медаль.
2. Для каждого участника (competitor\_id) подсчитывается количество таких записей.
3. Результат сохраняется во временную таблицу с колонками:
   * competitor\_id
   * total\_medals — общее количество медалей.

### 4️⃣ Удаление из временной таблицы

**Операция DELETE:**

1. Удаляются записи, где total\_medals = 0.  
   (Хотя по логике предыдущего шага таких быть не должно, так как фильтр medal\_id IS NOT NULL уже исключил их.)
2. Это может быть дополнительная проверка или универсальный шаг для очистки данных.

## 🔹 #2

### 1️⃣ Обновление таблицы person

**Цель:**  
Обновить значения роста (height) у каждой персоны, заменив их на среднее значение роста людей из того же региона.

**Логика:**

1. Подзапрос выбирает все записи p2 (других персон) из того же региона, что и текущая персона p.
2. Это достигается двойным соединением таблицы person\_region:
   * pr1 связывает текущую персону p
   * pr2 связывает других персон p2, принадлежащих тому же региону.
3. Затем для этого региона вычисляется средний рост AVG(p2.height).
4. Результат подставляется в обновляемое поле height текущей записи p.

### 2️⃣ Создание временной таблицы temp\_multievent

**Цель:**  
Найти участников, участвовавших более чем в одном событии в рамках одних Олимпийских игр.

**Логика:**

1. Соединяются games\_competitor и competitor\_event по идентификатору участника.  
   Это позволяет связать участника с событиями, в которых он участвовал.
2. Для каждой пары (competitor\_id, games\_id) подсчитывается количество событий.
3. HAVING COUNT(ce.event\_id) > 1 оставляет только тех, кто участвовал более чем в одном событии.
4. Результат сохраняется во временную таблицу.

### 3️⃣ Запрос по регионам с медалями

**Цель:**  
Определить регионы, средний показатель которых по медалям на участника выше общего среднего уровня.

**Пошагово:**

1. Извлекаются регионы (noc\_region), соединённые с людьми (person\_region), участниками (games\_competitor) и их событиями (competitor\_event).
2. Оставляются только случаи, где у события есть медаль (medal\_id IS NOT NULL).
3. Группировка выполняется по регионам.
4. В HAVING сравнивается среднее значение (численно выраженное через AVG(1), то есть количество участников с медалями) с общим средним количеством медалей на участника:
   * Подзапрос вычисляет среднее количество медалей на участника по всем спортсменам (через группировку по gc2.id).
   * Основной запрос выбирает только те регионы, где их локальное среднее выше этого общего среднего.

### 4️⃣ Создание временной таблицы temp\_season\_participation

**Цель:**  
Определить, в каких сезонах (Summer/Winter) участвовал каждый человек.

**Логика:**

1. Таблица games\_competitor соединяется с games, чтобы получить информацию о сезоне игр.
2. Сохраняются пары (person\_id, season) без дубликатов.
3. Это формирует таблицу участия в сезонах.

### 5️⃣ Финальный запрос

**Цель:**  
Найти людей, участвовавших **в обоих сезонах** — и летних, и зимних Олимпиадах.

**Логика:**

1. Группировка по person\_id.
2. HAVING COUNT(DISTINCT season) = 2 — значит, у человека есть участие как минимум в двух разных сезонах.

**Результат:**  
Список идентификаторов людей, участвовавших и в Summer, и в Winter Олимпийских играх.

## ✅ Итоговое резюме

| **№** | **Что делает блок** | **Основной результат** |
| --- | --- | --- |
| 1 | Средний возраст участников с медалями | Средний возраст по каждому типу медали и участнику |
| 2 | 5 регионов с наибольшим числом активных спортсменов | Список топ-5 регионов |
| 3 | Подсчёт количества медалей | Таблица участник–число медалей |
| 4 | Очистка временной таблицы | Исключение участников без медалей |
| 5 | Обновление роста по региональному среднему | Корректировка данных height |
| 6 | Участники с несколькими событиями | Временная таблица активных участников |
| 7 | Регионы с выше среднего количеством медалей | Аналитика успешных регионов |
| 8 | Таблица участия в сезонах | Персональные данные по сезонам |
| 9 | Участники двух сезонов | Список "универсальных" спортсменов |