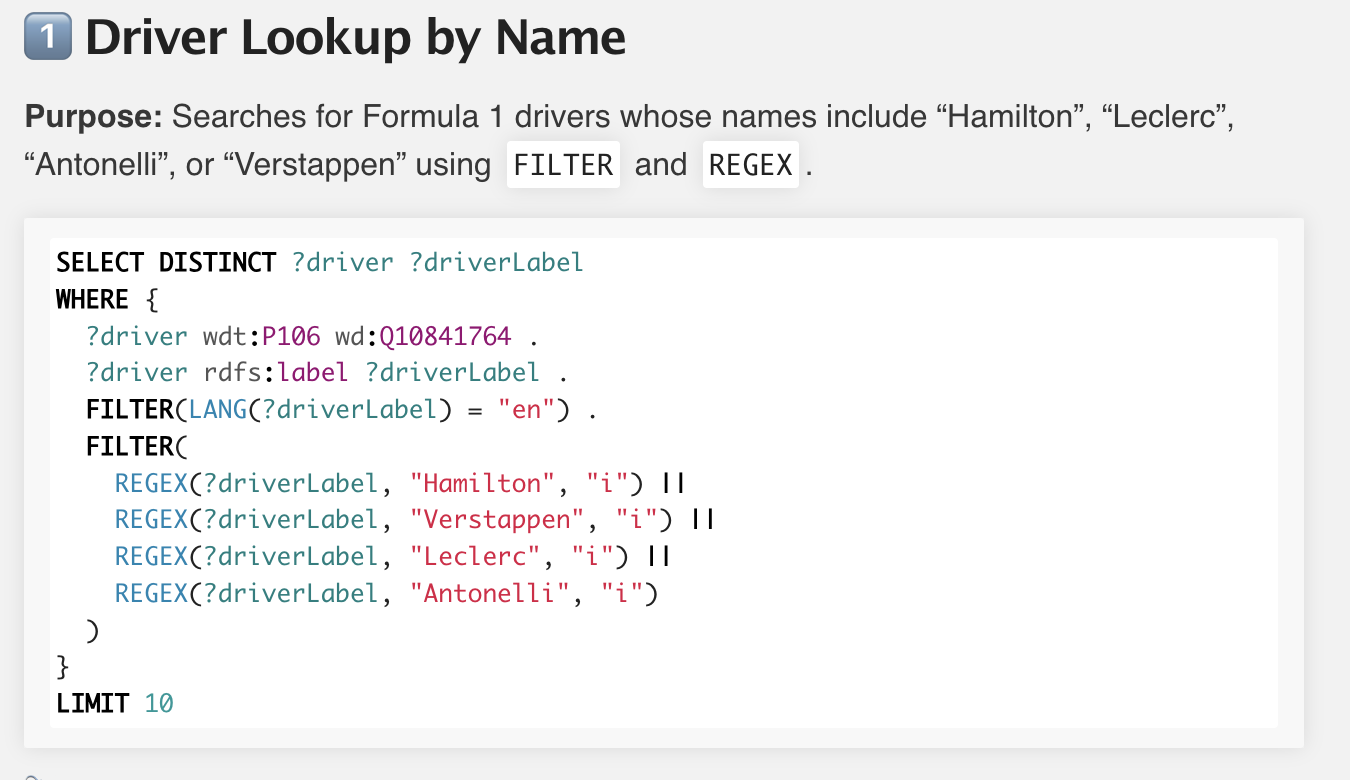
**SPARQL explanations**

****

**🔍 What does this SPARQL query do?**

**It searches for Formula 1 drivers** whose names include **“Hamilton”, “Verstappen”, “Leclerc”, or “Antonelli”**, using the FILTER and REGEX keywords.

We also filter for labels in **English** and use DISTINCT to remove duplicates.

**📘 How to explain each line step-by-step:**

**SELECT DISTINCT ?driver ?driverLabel**

This means:

“Give me a list of all drivers and their names — without duplicates.”

**WHERE {**

We start the query block here.

**?driver wdt:P106 wd:Q10841764 .**

We say:

“Only get results where the person’s **occupation** is Formula 1 driver.”

* P106 = "occupation"
* Q10841764 = "Formula One driver" in Wikidata

This limits the results only to F1 drivers.

**?driver rdfs:label ?driverLabel .**

We say:

“Also return the name (label) of each driver.”

rdfs:label gives us the human-readable name like “Lewis Hamilton”.

**FILTER(LANG(?driverLabel) = "en")**

We only want English names.

Without this, you could get labels in many languages — like French, German, etc.

**FILTER( REGEX(...))**

This is where we **search for specific names**.

We use the REGEX function to match names that:

* contain “Hamilton”
* or “Verstappen”
* or “Leclerc”
* or “Antonelli”

Each one is **case-insensitive** because of "i" (ignore case).

This part says:

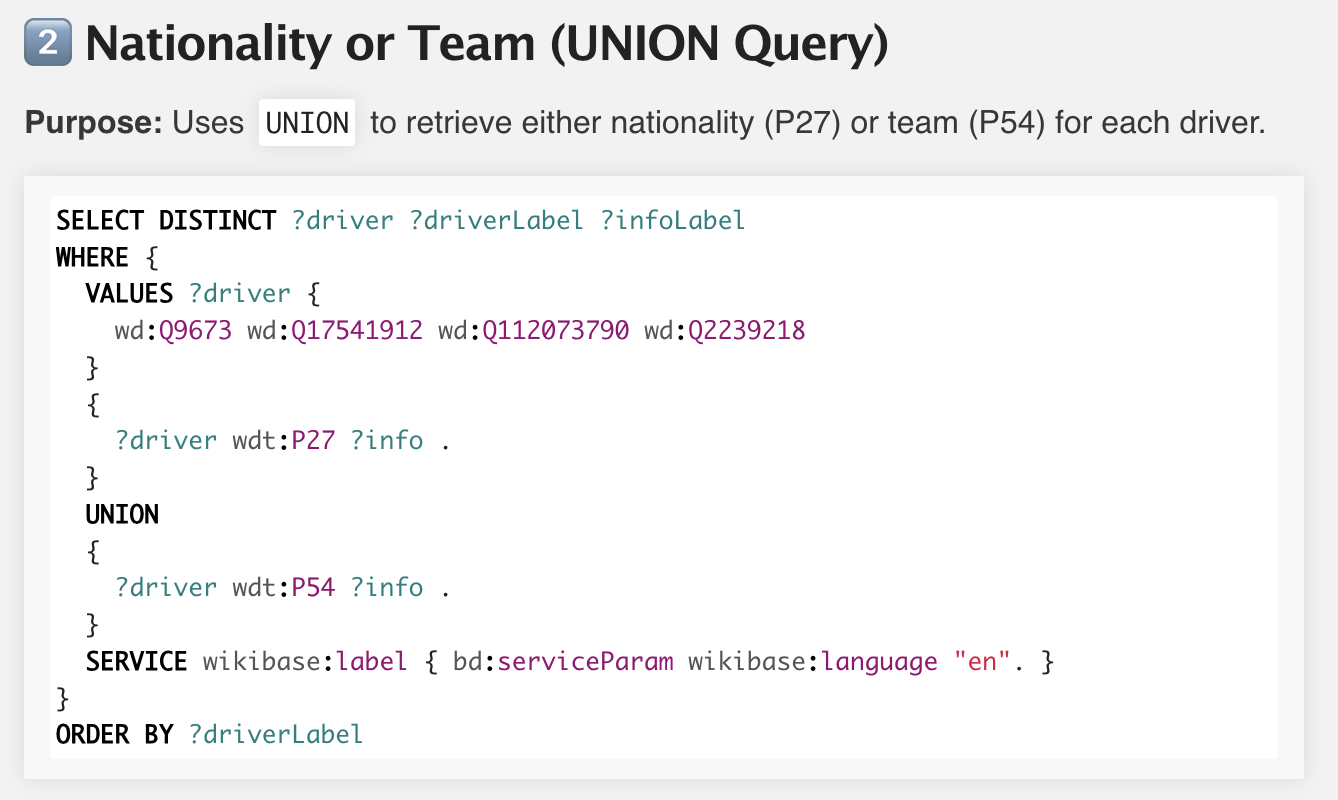
“Only return drivers whose name includes one of these 4 names.”

**LIMIT 10**

We only want the **first 10 results**, to keep the query fast.

**💬 How to say it in 1–2 sentences on exam:**

This SPARQL query searches for Formula 1 drivers by checking if their names match one of four values: Hamilton, Verstappen, Leclerc, or Antonelli.  
We use REGEX inside a FILTER block to match these names, and we also filter for English labels. The DISTINCT makes sure we don’t get duplicates, and LIMIT 10 keeps the results short.

****

**🔍 What does this SPARQL query do?**

This query gets **either the nationality or the team** of each selected driver — using the UNION keyword to combine two types of facts.

It shows how we can get **two different properties** in one query:

* P27 → nationality
* P54 → member of sports team

**📘 Step-by-step explanation:**

**SELECT DISTINCT ?driver ?driverLabel ?infoLabel**

Return a list of drivers, their names, and either their country or their team.  
We use DISTINCT to avoid duplicates.

**VALUES ?driver { ... }**

This block **manually selects the drivers** we want to check.

It includes:

* Q9673 = Lewis Hamilton
* Q17541912 = Charles Leclerc
* Q112073790 = Kimi Antonelli
* Q2239218 = Max Verstappen

This is like saying:

“We only want results for these four drivers.”

**{ ?driver wdt:P27 ?info . }**

This part says:

“Get the driver's **nationality**.”

* P27 is the Wikidata property for **country of citizenship**

**UNION { ?driver wdt:P54 ?info . }**

This says:

“Or get the driver's **team**.”

* P54 is the property for **member of sports team**

So the full UNION block means:

“Get either nationality OR team for each driver.”

**SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }**

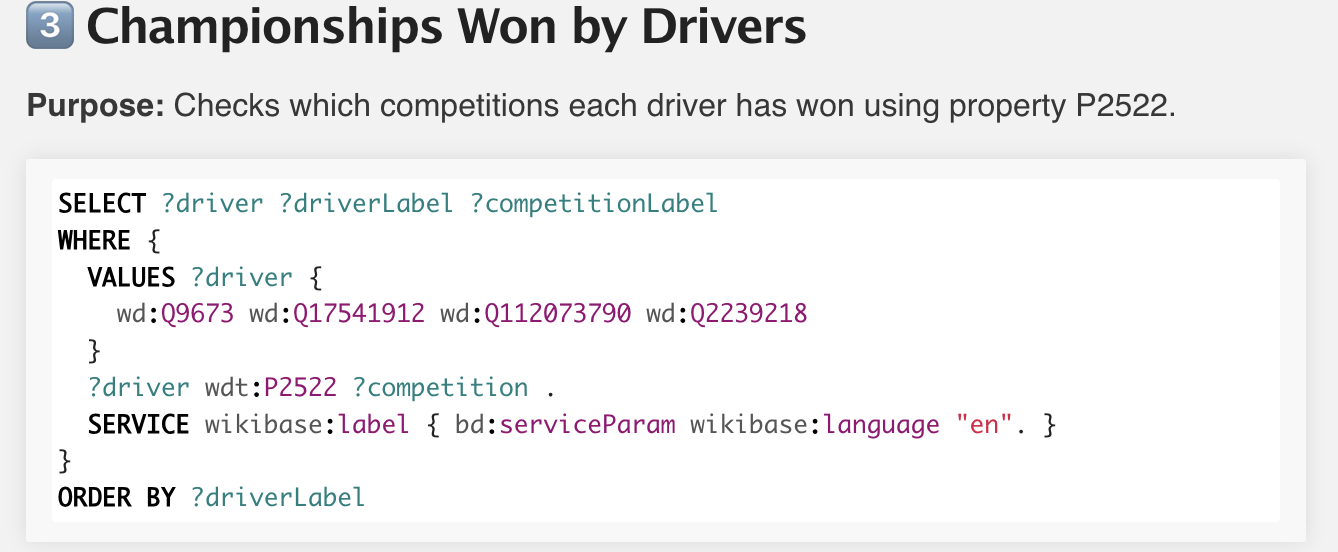
This part **adds human-readable labels** (like “Ferrari” instead of Q-ID).  
We specify that we want labels in English.

**ORDER BY ?driverLabel**

This sorts the results alphabetically by the driver’s name.

**💬 How to say it in 1–2 sentences:**

This query retrieves either the **nationality or team** for each driver using UNION.  
We manually select the drivers using VALUES, and then combine the results of two different properties: P27 for nationality and P54 for team.  
We also use SERVICE to get English labels and ORDER BY to sort the results by name.

****

**🔍 What does this query do?**

This query **checks which championships each selected driver has won** using the property P2522, which stands for **“competition won”** in Wikidata.

It gives us a list of:

* the driver
* the name of the driver
* the name of the championship they won (for example: “2021 Formula One World Championship”)

**📘 Step-by-step explanation:**

**SELECT ?driver ?driverLabel ?competitionLabel**

We ask for:

* the driver entity (?driver)
* their name (?driverLabel)
* and the competition they won (?competitionLabel)

**VALUES ?driver { ... }**

We manually pick the 4 drivers we want:

* Hamilton, Leclerc, Antonelli, Verstappen

This tells Wikidata:

“Only return results for these specific people.”

**?driver wdt:P2522 ?competition .**

This is the main part:

“Give me all **competitions** that the driver has won.”

* P2522 = "competition won"

So this will show if the driver has won a championship like:

* “2021 Formula One World Championship”
* “2023 F1 World Championship”, etc.

**SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }**

As before, this block is used to:

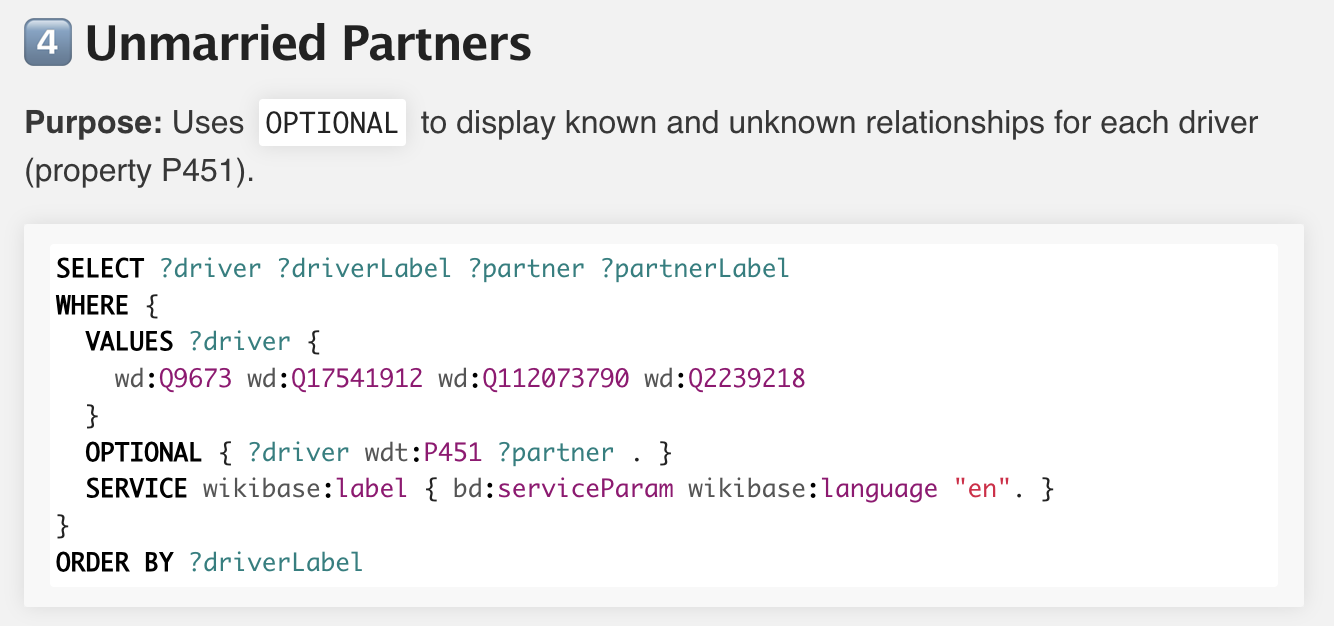
“Get readable labels for both the driver and the competition — in English.”

**ORDER BY ?driverLabel**

Sort results by name of the driver (A–Z).

**💬 How to say it in 1–2 sentences at the exam:**

This query checks which **Formula 1 championships** each selected driver has won using P2522, the property for “competition won.”  
We selected 4 drivers manually, used SERVICE to get labels in English, and sorted the results alphabetically.

****

**🔍 What does this query do?**

This query shows whether each selected driver has a known **unmarried partner** in Wikidata, using property P451.  
If there is **no partner listed**, the result will still include the driver — just with an empty value.

**📘 Step-by-step explanation:**

**SELECT ?driver ?driverLabel ?partner ?partnerLabel**

We ask for:

* the driver (Q-ID)
* the driver’s name
* the partner (if available)
* and the partner’s name

**VALUES ?driver { ... }**

We select only 4 drivers:

* Hamilton, Leclerc, Antonelli, Verstappen

**OPTIONAL { ?driver wdt:P451 ?partner . }**

This is the key part. P451 = **unmarried partner**.

Using OPTIONAL means:

“If there is a known partner, show it.  
If not — still show the driver.”

So, **no data ≠ no result** — that’s the purpose of OPTIONAL.

**SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }**

As before, we get readable names for both the driver and the partner — in English.

**ORDER BY ?driverLabel**

Results are sorted by driver name.

**💬 How to explain it in 1–2 sentences:**

This query checks which drivers have an **unmarried partner** using P451, but uses the OPTIONAL keyword.  
That way, even if a driver has no relationship listed in Wikidata, they are still shown in the results.

**📌 Example you can add if asked:**

For example, Hamilton and Verstappen had known partners, but **Charles Leclerc’s relationship** was missing — even though it’s public.  
This helped us detect that gap and include it in our enrichment.

**PROMPTING TECHNIQUES**

**1️. Zero-shot prompting**

❓ What is it?

You just ask a direct question — **no examples, no steps.**

🛠 Used for:

* Simple, well-known facts
* Example:

*“What team is Lewis Hamilton racing for in 2025?”*

✅ Models answered: **Ferrari**

📌 Summary phrase:

“Zero-shot is useful for clear, public facts that don’t need explanation.”

**2️. Few-shot prompting**

❓ What is it?

You give **examples first**, then ask the model to follow the same pattern.

🛠 Used for:

* **Less famous drivers** like Antonelli
* Example prompt:

Hamilton → Ferrari  
Verstappen → Red Bull  
Antonelli → ?

✅ Models answered: **Mercedes**

📌 Summary phrase:

“Few-shot works best when we guide the model with patterns — especially for low-data cases.”

**3️. Chain-of-Thought (CoT) prompting**

❓ What is it?

You ask the model to **reason step by step** to reach the answer.

🛠 Used for:

* **Recent or complex facts**
* Example:

*“Who won the 2024 F1 Championship? Let’s think step by step.”*

✅ Models answered: **Verstappen**, with detailed explanation (especially GPT)

📌 Summary phrase:

“Chain-of-Thought is best when the answer needs multi-step reasoning or context.”

**💬 Final sentence to use at exam:**

We used three prompting styles depending on the case:  
**Zero-shot** for simple facts, **few-shot** for low-visibility drivers like Antonelli, and **Chain-of-Thought** for complex or recent events like Verstappen’s 2024 title.

**RDF Triples**

****

**🧱 Твой RDF Triple:**

turtle

CopyEdit

wd:Q9673 wdt:P54 wd:Q169898 .

**💬 Человеческий перевод:**

Lewis Hamilton is a **member of** Scuderia Ferrari.

**🔍 Подробное объяснение:**

**1. wd:Q9673**

→ это **Lewis Hamilton**  
(все Q-коды — это сущности, то есть "кто" или "что")

**2. wdt:P54**

→ это свойство **“member of sports team”**  
(всегда означает принадлежность к команде, клубу и т.д.)

P54 — это **основное свойство**, без qualifiers  
(просто факт, без дополнительных деталей)

**3. wd:Q169898**

→ это **Scuderia Ferrari**  
(тоже сущность в Wikidata, идентификатор команды)

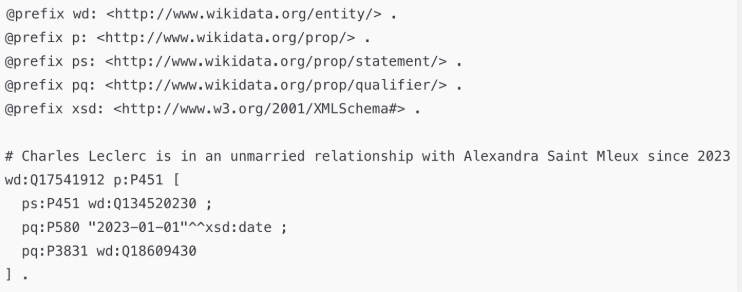
**⚠️ Что НЕ указано здесь:**

* Нет **start date** (когда он вступил в команду)
* Нет **sport type** (что это именно Formula 1)
* Нет **qualifiers** — например: сезон, контракт

Это всё добавляется через p:, ps:, pq: — если ты хочешь детализировать.

**💬 Как ответить на экзамене (готовая речь):**

This triple shows that **Lewis Hamilton is a member of Scuderia Ferrari**, using property P54, which means “member of sports team.”  
Q9673 is Hamilton, Q169898 is Ferrari, and P54 connects them.  
This is a basic triple — it gives us the fact, but **without qualifiers like start date or contract details.**

****

**🧱 Что говорит этот triple?**

turtle

CopyEdit

# Charles Leclerc is in an unmarried relationship with Alexandra Saint Mleux since 2023

📘 Перевод:

Шарль Леклер состоит в **незарегистрированных отношениях** с Александрой Сент-Млё с 2023 года.

**📘 Полный разбор**

**🔹 wd:Q17541912**

→ это **Charles Leclerc** (субъект)

**🔹 p:P451**

→ это связка для свойства **"unmarried partner"**  
(тут используется p: → потому что мы добавим qualifiers)

Внутри блока [ ... ]:

**🔹 ps:P451 wd:Q134520230 ;**

→ основное утверждение: **Александра Saint Mleux** — его партнёр

* ps: = "predicate statement"
* P451 = "unmarried partner"

**🔸 pq:P580 "2023-01-01"^^xsd:date ;**

→ qualifier: **с какой даты** они в отношениях

* P580 = "start time"

📌 Эта строка добавляет **контекст к факту**

**🔸 pq:P3831 wd:Q18609430**

→ qualifier: **тип отношений**

* P3831 = "relationship type"
* Q18609430 = "unmarried partner"

📌 Это уточнение, что это **не брак**, а именно незарегистрированные отношения

**💬 Как ответить на экзамене:**

This RDF triple shows that **Charles Leclerc is in an unmarried relationship** with **Alexandra Saint Mleux** since **January 2023**.  
We use p:P451 to open a block with qualifiers.  
ps:P451 gives the partner, pq:P580 gives the start date, and pq:P3831 defines the relationship type.  
This kind of triple is richer than a basic one — because it adds context and time qualifiers.

****

## 🏁 Что говорит этот triple?

turtle

CopyEdit

# Max Verstappen won the 2024 F1 World Championship, clinched at Las Vegas Grand Prix

📘 По-человечески:

Макс Ферстаппен выиграл **чемпионат мира по Формуле 1 в 2024 году**, и это произошло на **Гран-при Лас-Вегаса** — 24 ноября 2024 года.

## 📘 Подробный разбор

### 🔹 wd:Q2239218

→ Это **Макс Ферстаппен**

### 🔹 p:P2522

→ Это блок свойства **"competition won"**  
(то есть: какую важную награду он выиграл)

#### Внутри [ ... ]:

### 🔹 ps:P2522 wd:Q113628886 ;

→ Это **основной факт**:

“Он выиграл 2024 F1 World Championship”

### 🔸 pq:P585 "2024-11-24"^^xsd:date ;

→ Квалификатор даты (когда он стал чемпионом)

* P585 = “point in time”
* ^^xsd:date = техническая запись формата даты

📌 Это важно для современных фактов.

### 🔸 pq:P1344 wd:Q111443584

→ Это квалификатор **“event”**

* P1344 = “part of the event”
* Q111443584 = Las Vegas Grand Prix

📌 Мы говорим: “Он стал чемпионом именно на этой гонке”

## 💬 Как сказать это на экзамене:

This RDF triple shows that **Max Verstappen won the 2024 Formula One World Championship**, which is property P2522.  
The ps: statement gives us the championship, while the pq: qualifiers add the **date of the win (November 24)** and the **event where it happened (Las Vegas Grand Prix)**.  
It’s a perfect example of how **RDF triples + qualifiers** provide full context.

**Prefixes**

**The prefix section defines short names for longer URIs in RDF.**

* **wd: is for entities like people or teams**
* **wdt: is for direct properties like "team" or "nationality"**
* **ps: is for the main value inside a complex statement**
* **pq: is for qualifiers like date or event**
* **xsd: is used for data types like date and time**