EMNLP 2025
Suzhou, China | 中国苏州

Weaver: Interweaving SQL and LLM for Table Reasoning

EMNLP 2025

Rohit Khoja*, Devanshu Gupta*
Yanjie Fu, Dan Roth, Vivek Gupta

Arizona State University, University of Pennsylvania

Why is Table QA still Challenging?

- Tables contain both **structured** (numbers, fields) and **unstructured** (long text/images) data
- SQL is great for logic but fails at semantic inference
- LLMs handle semantics but struggle at structured logic

Example: “Which country had the most competitors?”

Driver	**Constructor**	**Laps**	**Time**
Alain Prost	Ferrari	64	1:18:31
Thierry Boutsen	Williams-Renault	64	39.092
Ayrton Senna	McLaren-Honda	63	1 Lap

SQL fails here → LLM helps with nationality inference

Existing SQL–LLM integration is rigid or shallow

Method	**Strength**	**Limitation**
Binder/BlendSQL	Integrate LLM into SQL	Fail on multi-step reasoning
H-STAR / ReAcTable	Structured pruning	Struggles with row extraction
ProTrix	2-step reasoning	Limited flexibility

Key Issue: Fixed workflows lack adaptability to complex queries

Weaver dynamically interweaves SQL and LLM reasoning

LLM-generated dynamic execution plan:

Weaver first generates a **flexible step-by-step plan** that adapts to query complexity, then executes through dynamic interweaving of:

1. **SQL step** → Structured operations (filter, aggregate, join)
2. **LLM step** → Semantic reasoning (inference, understanding)
3. **Verification** → Ensures correctness

Back-and-forth reasoning:
SQL ↔ LLM ↔ SQL ↔ LLM

Phase 1: Preprocessing

Prepare the data:

- Extract metadata and constraints
- Identify table schema and data types
- Filter irrelevant columns

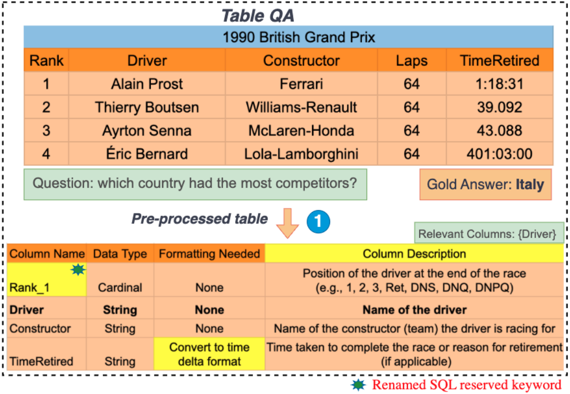


Table QA

1990 British Grand Prix				
Rank	Driver	Constructor	Laps	TimeRetired
1	Alain Prost	Ferrari	64	1:18:31
2	Thierry Boutsen	Williams-Renault	64	39.092
3	Ayrton Senna	McLaren-Honda	64	43.088
4	Éric Bernard	Lola-Lamborghini	64	401:03:00

Question: which country had the most competitors?

Gold Answer: **Italy**

Pre-processed table 1

Relevant Columns: {Driver}

Column Name	Data Type	Formatting Needed	Column Description
✸ Rank_1	Cardinal	None	Position of the driver at the end of the race (e.g., 1, 2, 3, Ret, DNS, DNQ, DNPQ)
Driver	**String**	**None**	**Name of the driver**
Constructor	String	None	Name of the constructor (team) the driver is racing for
TimeRetired	String	Convert to time delta format	Time taken to complete the race or reason for retirement (if applicable)

✸ Renamed SQL reserved keyword

Phase 2: Planning

LLM generates a dynamic execution plan:

- Generate step-by-step execution plan
- Determine SQL vs. LLM operations for each step
- Adapt plan based on query complexity

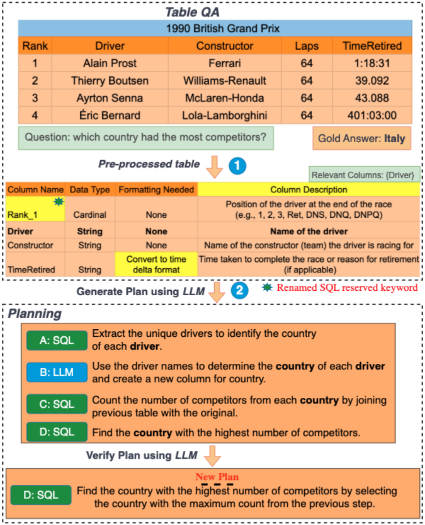


Table QA

1990 British Grand Prix				
Rank	Driver	Constructor	Laps	TimeRetired
1	Alain Prost	Ferrari	64	1:18:31
2	Thierry Boutsen	Williams-Renault	64	39.092
3	Ayrton Senna	McLaren-Honda	64	43.088
4	Éric Bernard	Lola-Lamborghini	64	401:03:00

Question: which country had the most competitors?

Gold Answer: **Italy**

Pre-processed table 1

Relevant Columns: {Driver}

Column Name	Data Type	Formatting Needed	Column Description
Rank_1 ✹	Cardinal	None	Position of the driver at the end of the race (e.g., 1, 2, 3, Ret, DNS, DNQ, DNPQ)
Driver	**String**	**None**	**Name of the driver**
Constructor	String	None	Name of the constructor (team) the driver is racing for
TimeRetired	String	Convert to time delta format	Time taken to complete the race or reason for retirement (if applicable)

Generate Plan using *LLM* 2

✹ Renamed SQL reserved keyword

Planning

A: SQL — Extract the unique drivers to identify the country of each **driver**.

B: LLM — Use the driver names to determine the **country** of each **driver** and create a new column for country.

C: SQL — Count the number of competitors from each **country** by joining previous table with the original.

D: SQL — Find the **country** with the highest number of competitors.

Verify Plan using *LLM*

New Plan

D: SQL — Find the country with the highest number of competitors by selecting the country with the maximum count from the previous step.

Phase 3: Code Execution

Dynamic interweaving of SQL and LLM:

- Execute SQL queries on structured data
- Run LLM inference for semantic tasks

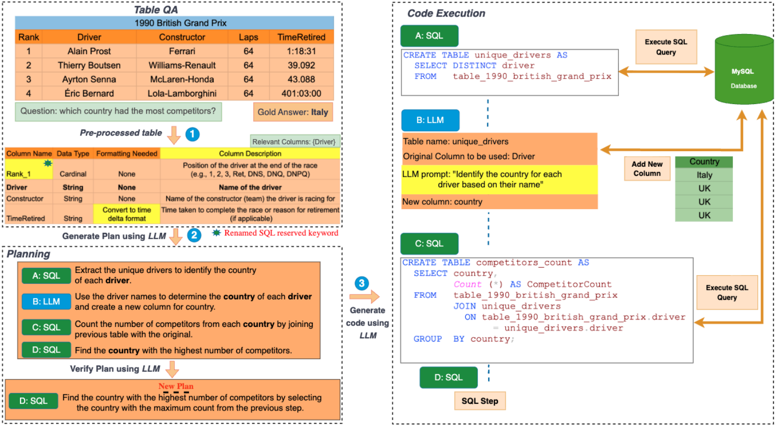


Table QA

1990 British Grand Prix				
Rank	Driver	Constructor	Laps	TimeRetired
1	Alain Prost	Ferrari	64	1:18:31
2	Thierry Boutsen	Williams-Renault	64	39.092
3	Ayrton Senna	McLaren-Honda	64	43.088
4	Éric Bernard	Lola-Lamborghini	64	401:03:00

Question: which country had the most competitors?

Gold Answer: **Italy**

Relevant Columns: {Driver}

Column Name	Data Type	Formatting Needed	Column Description
Rank_1 ✸	Cardinal	None	Position of the driver at the end of the race (e.g., 1, 2, 3, Ret, DNS, DNQ, DNPQ)
Driver	**String**	**None**	**Name of the driver**
Constructor	String	None	Name of the constructor (team) the driver is racing for
TimeRetired	String	Convert to time delta format	Time taken to complete the race or reason for retirement (if applicable)

✸ Renamed SQL reserved keyword

Planning

- A: SQL — Extract the unique drivers to identify the country of each **driver**.
- B: LLM — Use the driver names to determine the **country** of each **driver** and create a new column for country.
- C: SQL — Count the number of competitors from each **country** by joining previous table with the original.
- D: SQL — Find the **country** with the highest number of competitors.

New Plan

- D: SQL — Find the country with the highest number of competitors by selecting the country with the maximum count from the previous step.

Code Execution

A: SQL

```sql
CREATE TABLE unique_drivers AS
  SELECT DISTINCT driver
  FROM   table_1990_british_grand_prix
```

B: LLM

Table name: unique_drivers

Original Column to be used: Driver

LLM prompt: "Identify the country for each driver based on their name"

New column: country

Country
Italy
UK
UK
UK

C: SQL

```sql
CREATE TABLE competitors_count AS
  SELECT country,
         Count (*) AS CompetitorCount
  FROM   table_1990_british_grand_prix
         JOIN unique_drivers
           ON table_1990_british_grand_prix.driver
              = unique_drivers.driver
  GROUP  BY country;
```

D: SQL

Phase 4: Answer Extraction

Generate final answer:

- Extract Answer from final table
- Format and validate the final answer

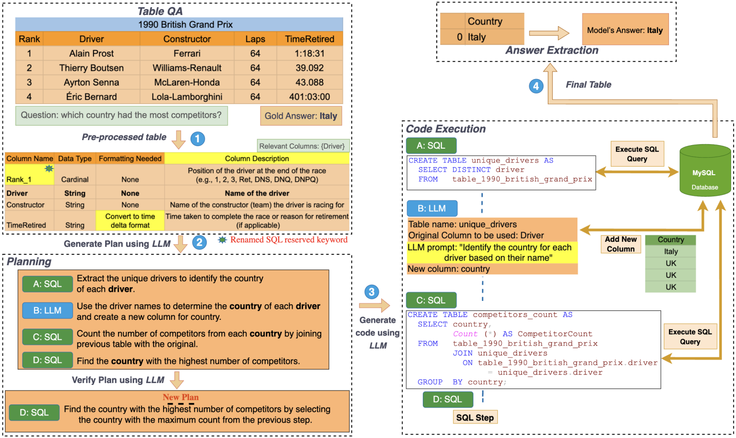


Table QA

1990 British Grand Prix				
Rank	Driver	Constructor	Laps	TimeRetired
1	Alain Prost	Ferrari	64	1:18:31
2	Thierry Boutsen	Williams-Renault	64	39.092
3	Ayrton Senna	McLaren-Honda	64	43.088
4	Éric Bernard	Lola-Lamborghini	64	401:03:00

Question: which country had the most competitors?

Gold Answer: **Italy**

Pre-processed table 1

Relevant Columns: {Driver}

Column Name	Data Type	Formatting Needed	Column Description
Rank_1 ✸	Cardinal	None	Position of the driver at the end of the race (e.g., 1, 2, 3, Ret, DNS, DNQ, DNPQ)
Driver	**String**	**None**	**Name of the driver**
Constructor	String	None	Name of the constructor (team) the driver is racing for
TimeRetired	String	Convert to time delta format	Time taken to complete the race or reason for retirement (if applicable)

✸ Renamed SQL reserved keyword

Generate Plan using *LLM* 2

Planning

- A: SQL — Extract the unique drivers to identify the country of each **driver**.
- B: LLM — Use the driver names to determine the **country** of each **driver** and create a new column for country.
- C: SQL — Count the number of competitors from each **country** by joining previous table with the original.
- D: SQL — Find the **country** with the highest number of competitors.

Verify Plan using *LLM*

New Plan

- D: SQL — Find the country with the highest number of competitors by selecting the country with the maximum count from the previous step.

3 Generate code using *LLM*

Code Execution

A: SQL

```sql
CREATE TABLE unique_drivers AS
  SELECT DISTINCT driver
  FROM   table_1990_british_grand_prix
```

B: LLM

Table name: unique_drivers
Original Column to be used: Driver
LLM prompt: "Identify the country for each driver based on their name"
New column: country

Country
Italy
UK
UK
UK

C: SQL

```sql
CREATE TABLE competitors_count AS
  SELECT country,
         Count (*) AS CompetitorCount
  FROM   table_1990_british_grand_prix
         JOIN unique_drivers
           ON table_1990_british_grand_prix.driver
              = unique_drivers.driver
  GROUP  BY country;
```

D: SQL

4 *Final Table*

	Country
0	Italy

Model's Answer: **Italy**

Answer Extraction

Example Walkthrough:

Question: Which country had the most competitors?

1. **SQL step:** Extract unique drivers

   `SELECT DISTINCT driver COUNT(*) FROM table`

2. **LLM step:** Infer country from `driver` column

   `"Alain Prost" → France, "Thierry Boutsen" → Belgium`

3. **SQL step:** Count competitors by country

   ```
   SELECT country, COUNT(*) as competitors
   FROM unique_drivers GROUP BY country
   ```

4. **Final Answer:** Italy

Key Benefit: Every step is transparent and interpretable

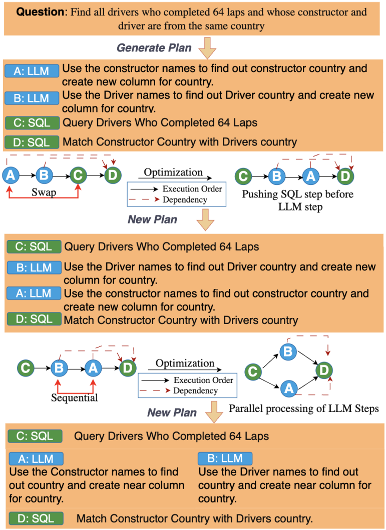
Planning Optimization for Fewer API Calls

Optimization strategies:

- **SQL reordering**
- **Parallelization**
- **Batch processing**

Result:

23% reduction in total steps with 1% accuracy loss



Question: Find all drivers who completed 64 laps and whose constructor and driver are from the same country

Generate Plan

A: LLM Use the constructor names to find out constructor country and create new column for country.

B: LLM Use the Driver names to find out Driver country and create new column for country.

C: SQL Query Drivers Who Completed 64 Laps

D: SQL Match Constructor Country with Drivers country

New Plan

C: SQL Query Drivers Who Completed 64 Laps

B: LLM Use the Driver names to find out Driver country and create new column for country.

A: LLM Use the constructor names to find out constructor country and create new column for country.

D: SQL Match Constructor Country with Drivers country

New Plan

C: SQL Query Drivers Who Completed 64 Laps

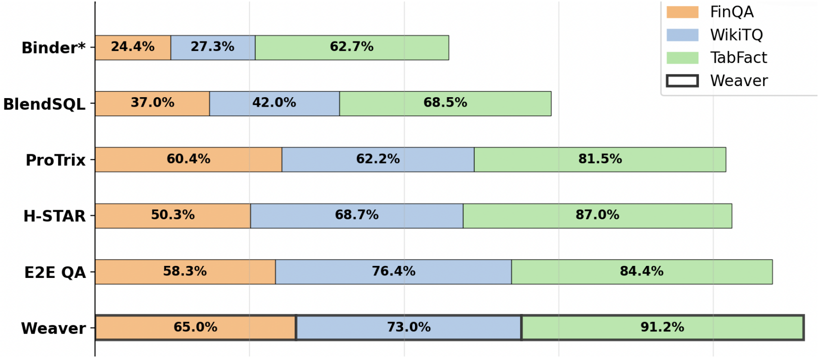
A: LLM Use the Constructor names to find out country and create near column for country.

B: LLM Use the Driver names to find out country and create near column for country.

D: SQL Match Constructor Country with Drivers country.

Weaver Outperforms State-of-the-Art

Performance on major benchmarks:



Key achievements:

- **+5%** accuracy improvement across datasets

Extends to Text + Image Tables

Multimodal Table QA Performance:

Dataset	**Modalities**	**Accuracy Gain**
MMTabQA	Text + Images	**+6.6%**
FinQA-MM	Tables + Passages	**+17.3%**
OTT-QA-MM	Tables + Passages	**+2.9%**

Highlight: Weaver handles reasoning across:

- Structured tables
- Unstructured text
- Embedded images

Unified framework for multimodal table reasoning

Efficacy & Efficiency

Efficiency:

- Average 6 API calls per query

Method	API Calls
Binder	50
H-STAR	8
Weaver	**5.5**

Efficacy:

- 28.1% accuracy improvement on large tables

Method	API Calls
H-STAR	35.9%
ProTrix	37.5%
Weaver	**65.6%**

Interpretability:

- Transparent step-by-step plan
- Intermediate tables visible
- Easy debugging and verification

Conclusion

Dynamic SQL–LLM weaving enables accurate, interpretable, and efficient Table QA

Key Takeaways:

- **Modular, interpretable pipeline** for hybrid table reasoning
- **5–10% accuracy gain** over state-of-the-art methods
- **Multimodal support** (text, image, table)
- **Flexible planning** adapts to query complexity

Link:

`coral-lab-asu.github.io/weaver`

Future Work

- Multi-table reasoning with joins across databases
- Multilingual table support (non-English tables)
- Hierarchical & nested data structures
- Integration with database systems