Learning to Extract Unique Rows in Google Sheets with the QUERY Function

Authored by

Mohammed looti

October 31, 2025

In the realm of Google Sheets, effective data management often hinges on the ability to handle and eliminate duplicate data. Whether your goal is generating comprehensive reports, ensuring database cleanliness, or preparing input for advanced analysis, extracting only the unique rows is a critical requirement for maintaining data integrity and maximizing operational efficiency. This comprehensive guide details the utilization of a highly effective technique: combining the inherent power of the UNIQUE function with the robust query capabilities of the QUERY function in Google Sheets to achieve precise, duplicate-free data extraction.

The fundamental strategy revolves around the concept of nesting, specifically encapsulating the UNIQUE function around the QUERY function. This methodological approach ensures a two-step data refinement process: first, the data is filtered and selected according to specific criteria using the `QUERY` function; subsequently, the `UNIQUE` function processes this result set to efficiently remove any redundant rows. This powerful combination guarantees that your final output is both highly relevant to your criteria and entirely free from duplication.

The standard syntax required for this indispensable operation is remarkably simple yet exceptionally effective for a wide variety of data manipulation tasks within your spreadsheet environment:

=UNIQUE(QUERY(A1:B16, "SELECT A, B"))

By wrapping the UNIQUE() function around the QUERY() function, you precisely dictate the execution flow to Google Sheets. The system first executes the inner query, which generates a preliminary, potentially duplicated result set. Immediately following this, the outer `UNIQUE()` function processes this intermediate output, swiftly identifying and eliminating any identical rows to present a final, clean output comprising only distinct entries. This foundation is key to advanced data handling.

To fully appreciate the efficiency of this strategy, we must delve into practical, real-world examples that illustrate how this powerful syntax can be utilized to significantly enhance your existing data manipulation and reporting capabilities.

Understanding the QUERY Function's Power

Before proceeding to the combined approach, it is essential to firmly grasp the sophisticated capabilities inherent in the QUERY function. Frequently cited as Google Sheets' equivalent to SQL (Structured Query Language), the `QUERY` function enables users to execute highly complex data manipulations. These manipulations include selecting specific data columns, filtering rows based on multiple conditions, aggregating numerical data, and meticulously sorting results. It stands as an incredibly versatile and indispensable instrument for extracting exactly the required subset of

information from a much larger dataset.

The standardized structure of the `QUERY` function is defined as `=QUERY(data, query, )`. Understanding each component is crucial for effective implementation:

`data` specifies the complete range of cells or array that you intend to query, typically defined using standard cell notation (e.g., `A1:D100`).

`query` is the core element, provided as a text string that contains the SQL-like commands (e.g., `"SELECT A, B WHERE C > 10"`). These commands dictate the exact selection, filtering, and grouping logic.

is an optional parameter used to specify the exact number of header rows present within your defined `data` range, which helps `QUERY` correctly interpret column types.

The unprecedented ability of `QUERY` to dynamically reshape and manipulate data based on user-defined criteria makes it a cornerstone function for advanced data analysis and automated reporting within the Google Sheets ecosystem. Mastering this function is key to elevating your spreadsheet proficiency beyond basic formulas.

Leveraging the UNIQUE Function for Data Integrity

In contrast to the complexity of `QUERY`, the UNIQUE function offers a streamlined, yet equally crucial, utility within Google Sheets: the precise elimination of redundant entries. When applied to any specified data range or array, `UNIQUE` executes one primary task--it returns only the distinct rows found in that input. This capability is exceptionally valuable for core functions such as rigorous data cleaning procedures, compiling definitive lists of non-repeating items, or ensuring that management reports are not skewed by misleading duplicate records.

The operational logic of the function relies entirely on row-wise uniqueness comparison. A row is only flagged and removed as a duplicate if, and only if, the values contained within all of its cells are an exact match to the corresponding values in another row. If even a minor disparity exists in a single cell across the compared rows, they are meticulously maintained and treated as distinct entries in the final output. This strict comparison mechanism guarantees the highest level of data fidelity.

While the `UNIQUE` function is undoubtedly powerful when used in isolation, its true analytical capacity, particularly for handling complex, large-scale data scenarios, is fully realized when it is strategically integrated and combined with other advanced functions, most notably the versatile `QUERY` function.

Combining UNIQUE and QUERY: The Core Strategy

The synergistic relationship established between the UNIQUE function and the QUERY function forms the backbone of this powerful technique for guaranteed unique row extraction. When the `QUERY` function is strategically embedded within the argument of `UNIQUE`, Google Sheets meticulously executes the operations in a defined, sequential order, ensuring controlled data transformation:

First, the inner QUERY function initiates the process. It accesses your defined data range and processes it strictly according to the SQL-like commands specified (e.g., selecting columns, applying filtering criteria). This initial step results in the creation of an intermediate result set, which, depending on the source data, is highly likely to contain duplicate rows.

Second, the outer UNIQUE function accepts this intermediate result set as its sole input. It then methodically scans every row within this input, performing the necessary row-wise comparisons and efficiently removing any exact duplicates. The final presentation to the user consists exclusively of the distinct rows that survived the filtering and cleaning process.

This methodical, two-stage execution process is crucial. It first allows the user to precisely define and retrieve **what** specific data elements are needed using the comprehensive selection and filtering capabilities of `QUERY`. Following this retrieval, the process guarantees that, from that targeted selection, **only unique entries** are presented, satisfying both the targeted retrieval and the strict data uniqueness requirements simultaneously. This is the cornerstone of advanced, reliable data reporting.

Practical Example: Extracting Unique Combinations

Let us apply this technique to a concrete, illustrative scenario. Imagine you are managing a sports dataset pertaining to 15 basketball players. This data includes their assigned Team (Column A) and their designated Position (Column B). In this scenario, it is highly probable that multiple players belong to the same team and occupy the same position, resulting in numerous redundant "Team-Position" combinations if your analytic focus is solely on unique pairings rather than individual player records.

The source data is organized efficiently in your sheet, with the key information residing in columns A (for "Team") and B (for "Position"), as depicted below:

	A	B	C	D
1	**Team**	**Position**		
2	A	Guard		
3	A	Guard		
4	A	Guard		
5	A	Forward		
6	A	Forward		
7	B	Guard		
8	B	Forward		
9	B	Forward		
10	B	Forward		
11	B	Center		
12	C	Guard		
13	C	Guard		
14	C	Guard		
15	C	Forward		
16	C	Center		
17				
18				

Our specific objective is to isolate and extract every unique pairing of **Team** and **Position** from the comprehensive source dataset. This means that if the combination "Team A, Guard" appears multiple times in the source data, the final result set must display it exactly once. We can accomplish this precision using the following nested formula:

=UNIQUE(QUERY(A1:B16, "SELECT A, B"))

Within this formula, the `QUERY` function first executes the internal command, selecting columns A (Team) and B (Position) from the defined range `A1:B16`. Crucially, the external `UNIQUE` function then processes the output generated by the query, rigorously ensuring that every resulting row represents a truly distinct combination of Team and Position before final display.

The subsequent screenshot visually confirms the application of this formula and showcases the concise, resulting list of unique combinations:

D1 =UNIQUE(QUERY(A1:B16,"SELECT A, B"))

	A	B	C	D	E
1	**Team**	**Position**		Team	Position
2	A	Guard		A	Guard
3	A	Guard		A	Forward
4	A	Guard		B	Guard
5	A	Forward		B	Forward
6	A	Forward		B	Center
7	B	Guard		C	Guard
8	B	Forward		C	Forward
9	B	Forward		C	Center
10	B	Forward			
11	B	Center			
12	C	Guard			
13	C	Guard			
14	C	Guard			
15	C	Forward			
16	C	Center			
17					
18					
19					

As is evident from the output, the resulting table displays only the unique pairings of **Team** and **Position**. For instance, regardless of how many individual player records exist in the original dataset where the **Team** is "A" and the **Position** is "Guard," this specific row combination appears only once in the final query result. This clearly demonstrates the efficacy of `UNIQUE` in robustly consolidating identical rows immediately following the initial targeted data selection executed by `QUERY`.

Enhancing Queries with WHERE Clauses and UNIQUE

The strategic advantage of nesting `UNIQUE` within `QUERY` is not restricted to simple column selection; it fully integrates with the complex capabilities of the QUERY function. This allows for the incorporation of advanced clauses, such as the WHERE clause, enabling you to first rigorously filter your data based on specific conditions and subsequently extract only the unique rows from that already-filtered subset. This dual-action approach facilitates highly targeted data extraction and comprehensive data cleaning operations simultaneously.

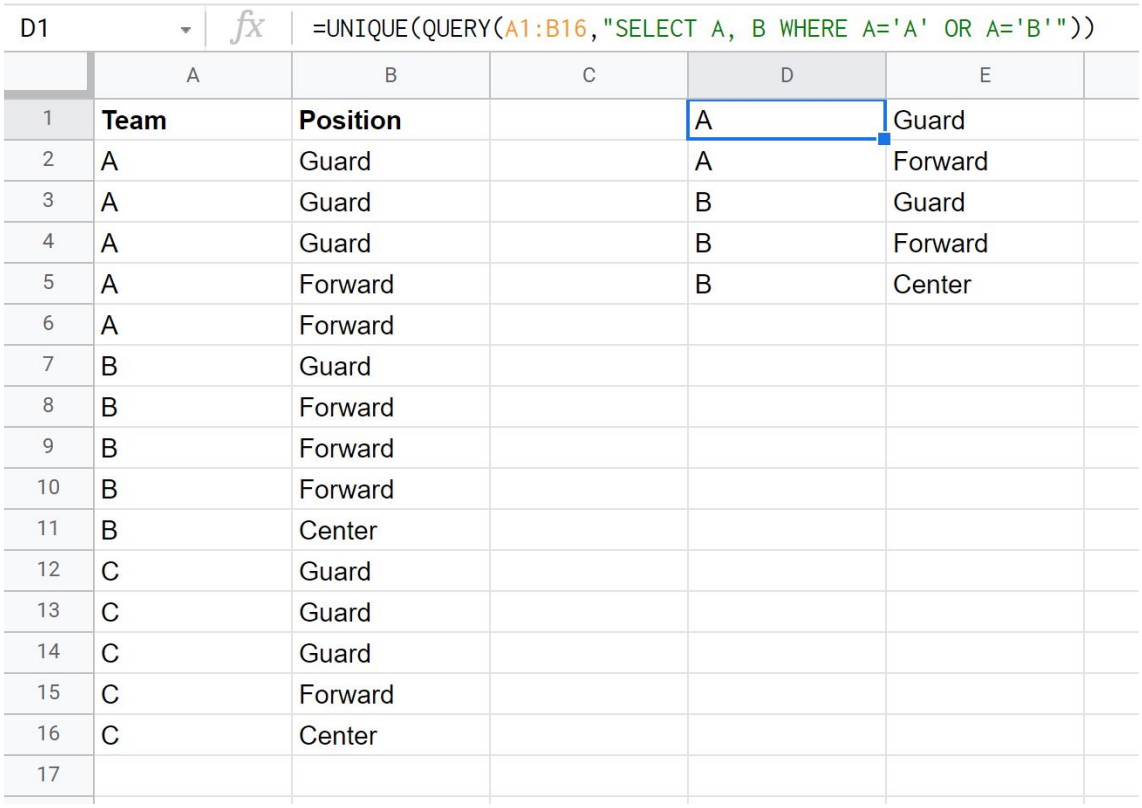
Consider a modification to our previous example: suppose the requirement is to retrieve unique Team and Position combinations, but strictly for players belonging to either Team "A" or Team "B". To satisfy this conditional requirement, we must modify our `QUERY` string to incorporate a

WHERE clause that precisely defines these filtering conditions:

=UNIQUE(QUERY(A1:B16, "SELECT A, B WHERE A='A' OR A='B'"))

In this advanced structure, the `QUERY` function first executes the internal logic, meticulously filtering the source dataset to include only those rows where column A (Team) matches 'A' or 'B'. Following this targeted selection, the `UNIQUE` function processes the filtered result set, ensuring that only distinct "Team-Position" combinations originating from the specified teams are ultimately returned to the user.

The final screenshot below demonstrates the concise output generated by this more sophisticated formula, showing unique rows strictly confined to teams A and B:



	A	B	C	D	E
1	**Team**	**Position**		A	Guard
2	A	Guard		A	Forward
3	A	Guard		B	Guard
4	A	Guard		B	Forward
5	A	Forward		B	Center
6	A	Forward			
7	B	Guard			
8	B	Forward			
9	B	Forward			
10	B	Forward			
11	B	Center			
12	C	Guard			
13	C	Guard			
14	C	Guard			
15	C	Forward			
16	C	Center			
17					

This result definitively confirms that the formula successfully integrated conditional filtering logic with robust duplicate elimination. The outcome is a clean and precise data output that adheres strictly to both the initial conditional criteria and the fundamental requirement for uniqueness across all returned rows.

Key Considerations and Best Practices

When implementing the powerful `UNIQUE(QUERY())` combination within Google Sheets,

adherence to certain best practices is crucial for optimizing performance, ensuring accuracy, and maintaining data reliability:

Performance on Large Datasets: While highly effective, querying extraordinarily large datasets (those containing potentially tens of thousands of rows or more) may sometimes introduce latency. For sheets of substantial size, it is prudent to explore preliminary data pre-filtering options or techniques to further optimize the internal structure of the query.

Defining "Unique" Precisely: It is paramount to remember that the `UNIQUE` function operates by identifying rows as duplicates only if **all** values across **all** columns specified in the query are perfectly identical. If your `SELECT` clause includes a greater number of columns, the criteria for a unique row becomes more stringent, which may consequently lead to a larger final result set.

The Importance of Column Selection: The specific columns chosen for inclusion in your `SELECT` clause directly determine the scope of what `UNIQUE` interprets as a "unique row." If you deliberately select only a subset of available columns, `UNIQUE` will evaluate uniqueness based exclusively on the data in those selected columns, completely disregarding any variation present in unselected columns.

Rigorous Error Handling: Before deploying complex formulas, always verify that your `QUERY` syntax is impeccable. Errors or typos within the `QUERY` string will cause the entire nested formula to fail. A reliable debugging technique involves running the QUERY function independently first to isolate and resolve errors, and only then wrapping it with `UNIQUE`.

Using Dynamic Ranges: To ensure that your `QUERY` remains adaptive and automatically incorporates new data as your dataset expands, it is highly recommended to define an open-ended range (e.g., specifying `A1:B` instead of the rigid `A1:B16`). This methodology guarantees that any new data appended to the bottom of the sheet is seamlessly integrated into the query's operational scope.

Conclusion and Further Reading

Mastering the combined `UNIQUE(QUERY())` function represents a significant milestone toward achieving advanced data management proficiency in Google Sheets. This technique provides a robust, scalable solution for extracting precise, highly filtered, and guaranteed duplicate-free information tailored exactly to your specific analytical needs. By cultivating a deep understanding of the individual strengths of both the `UNIQUE` and `QUERY` functions and recognizing how their complementary nature enhances data processing, you can dramatically elevate the sophistication and reliability of your data workflows.

For professionals seeking continued exploration and guidance on executing other common, high-

value operations within Google Sheets, we recommend consulting the following related tutorials and resources for further learning: