PROJECT REPORT

By:
Khushi Gandhi
(Group 83)
USC ID: 6401790989

Course Instructor: ***Dr. Wensheng Wu***

Table of Contents

I. Introduction

The objective of my database systems project is to implement a data management system with an emphasis on a sizable relational dataset. This dataset [1] contains observations of RF signals and includes a wide range of variables, such as Timestamp, Antenna Type, Temperature, etc.

This project centers on a custom database system that can handle different datasets. I have tested its possibilities using this 494MB relational dataset. Inspired by MySQL's tabular layout, the system efficiently arranges rows and columns to offer a robust framework for handling RF signal data as well as other desired datasets.

To improve user engagement, I have implemented a unique query language that is like MySQL and allows users to easily explore the information. For example, natural language queries such as "arrange the data in IQData by DeviceType with ascending order" are used. A friendly command-line interface (CLI) enhances the query language and promotes easy system interactions. Users may issue instructions and explore the possibilities of the system with ease with the familiar 'Enter Query >' prompt.

I will go over the complexities of the system's architecture and implementation in this report, demonstrating its usefulness by using the relational dataset to illustrate how well queries and data updates work.

II. Planned Implementation

a. Table Overview

Initially, I had a plan to create specific tables for testing my custom query language. The tables were meant to cover different aspects of RF signal observations:

1. **SignalData:** Timestamp, Frequency, Signal Strength, Modulation, Device Type, Antenna Type.

2. **Weather:** Timestamp, Temperature, Humidity, Wind Speed, Precipitation, Weather Condition.

3. **DeviceStatus:** Timestamp, Interference Type, Battery Level, Power Source, CPU Usage, Memory Usage, WiFi Strength, Disk Usage, System Load.

4. **Geolocation:** Timestamp, Latitude, Longitude.

However, after looking closely at the dataset, I realized some columns were redundant. So, I made adjustments to simplify the structure:

1. **SignalData:** Timestamp, Frequency, SignalStrength, Modulation, DeviceType, AntennaType.

2. **Weather:** Timestamp, Temperature, Humidity, WindSpeed, Precipitation, WeatherCondition.

3. **Status:** Timestamp, InterferenceType, CPUUsage, MemoryUsage, WiFiStrength, DiskUsage, SystemLoad, DeviceStatus.

4. **IQData:** Timestamp, DeviceType, IQData.

b. Query Language Overview

Initially, I planned to develop the query language using Python and ANTLR, aiming to incorporate commands for creating tables, inserting data, filtering, ordering, updating, and deleting data. [2]
However, as detailed further in the document due to encountered challenges, I eventually crafted the language in Python without relying on ANTLR. Here's a simplified example of how the query language might look:

1. **Projection (Selecting a subset of rows):**
   `display <column> from <table>`
   Example: **display AntennaType, DeviceType from SignalData**

2. **Filtering (Selecting rows):**
   `give the entries from <column> in <table> where <conditions>`
   Example: **give the entries from Weather where WindSpeed > 10**

3. **Join (Combining multiple data tables):**
   `combine <table 1> and <table 2> using <common key> with <join type>`
   Example: **combine SignalData and IQData using DeviceType with inner**

4. **Grouping:**
   `group <columns> from <table> with <other column> using <aggregation function>.`
   Example: **group WeatherCondition from Weather with Humidity using sum**

5. **Aggregation:**
   `compute <columns> from <table> using <function>`
   Example: **compute Temperature, Humidity from Weather using count**

6. **Ordering:**
   `arrange the data <table> by <column> with <ascending/descending> order`
   Example: **arrange the data in IQData by DeviceType with ascending order**

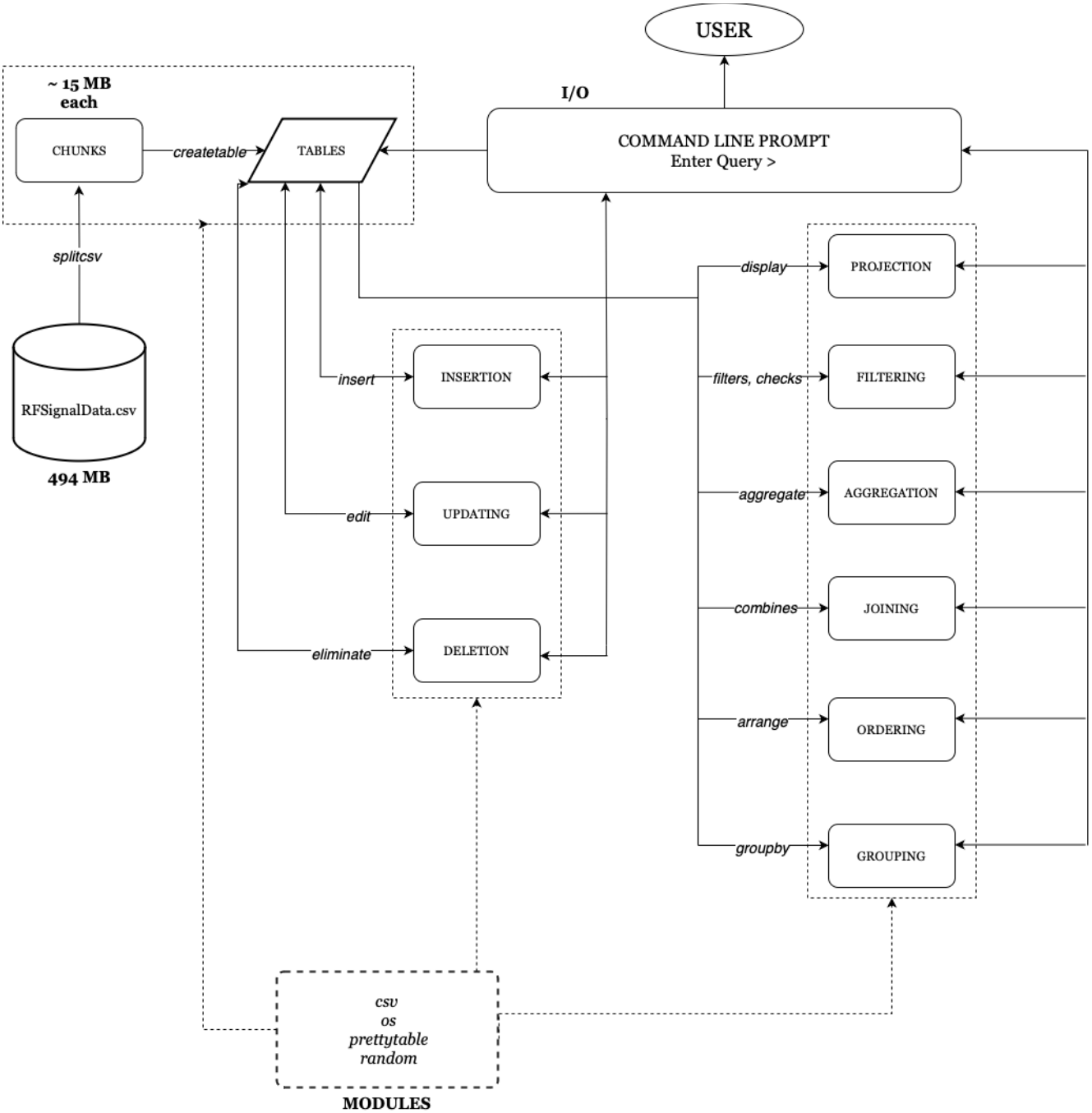
7. **Inserting Data:**
   `include a new entry in <table> with <values>`
   Example: **include a new entry in Weather with Timestamp == 6/5/23 09:00, Temperature == 35, Humidity == 40, WindSpeed == 23, Precipitation == 0, WeatherCondition == Sunny**

8. **Updating Data:**
   `edit <column> to <updated value> where <condition> in <table>`
   Example: **edit InterferenceType to NA where InterferenceType == 'None' in Status**

9. **Deleting Data:**
   `eliminate from <table> where <conditions>`
   Example: **eliminate from DeviceStatus where MemoryUsage == '0'**

10. **Creating Table:**
    `produce table <table> with columns: <column 1> <dtype, <column 2> <dtype>`
    Example: **produce table IQData with columns: Timestamp datetime, DeviceType string, IQData list**

III. Architectural Design



The above flowchart depicts the **Input, Output, Data Flow and Modules** used for the project. *The text on the arrows shows the function I have created to make each query using Python.*

[The chunk size mentioned is what I have used here while testing, but it can be changed as per the user and system requirement.]

IV. Implementation

a. Functionalities [2]

I. Chunking

The **splitcsv** function serves the purpose of implementing a manual chunking process for the RF signal dataset in your project. The objective is to adhere to the project guideline that explicitly states the system should not load the entire database into memory but should be able to handle potentially large datasets that might not fit into the main memory. The code takes as input the path to the RF signal dataset in CSV format (**RFSignalData.csv**), the output folder path where the dataset chunks will be stored, and a maximum memory constraint (**maxmem**). The primary goal is to split the dataset into smaller, more manageable chunks, thus facilitating more efficient processing.

Upon initialization, the code sets up essential variables, including **chunksize** to determine the number of rows in each chunk, **header** to store the CSV file header, and **current** to keep track of the current chunk number. The code then enters a loop to read and process chunks of data. It reads rows from the CSV file until the memory usage of the current chunk exceeds the specified **maxmem** or the number of rows reaches the defined **chunksize**. The data for the chunk is stored in the **data** list.

For every chunk, provided that the data list is not empty, it proceeds to record the segment data into a newly created CSV file within the designated output directory. Every chunk is stored in an individual CSV file, conforming to a naming pattern like **RFSignalData_1.csv, RFSignalData_2.csv,** and so on. Afterwards, the present variable is incremented to monitor the chunk number. To guarantee correct implementation, the code verifies the presence of the output folder "**Chunks**" and generates it if it is not already present.

The method of chunking allows the system to operate on smaller segments of the dataset, in accordance with the principle of handling potentially enormous datasets without putting the entire database into memory.

II. Creating a Table

produce table <table> with columns: <column 1> <dtype, <column 2> <dtype>

The **createtable** function is crucial for dynamically constructing tables in response to the user's query. The process starts by selecting a chunk file at random from the ones generated throughout the dataset chunking procedure. The arbitrary selection guarantees the inclusiveness of data for evaluating query operations. The function carefully analyzes the user's query to extract crucial details, including the requested table name, column names, and their corresponding data types. Afterwards, it verifies whether the given columns match the header of the chosen chunk file, guaranteeing uniformity in the format. The function beautifully presents the table using the PrettyTable library, ensuring that the columns are aligned to the left for enhanced reading. Meanwhile, the data is saved in the 'Tables' folder as a CSV file, which allows for convenient retrieval and additional examination.

III. Projection

display <column> from <table>

The purpose of the **display** function is to project specific columns from a selected table. The system skillfully analyzes the user's query to extract the desired columns and the corresponding table name. This function plays a crucial role in enabling the user to explore specific subsets of data. The function verifies the presence of the provided table to ensure that the requested information is accessible for presentation. By utilizing PrettyTable, the chosen columns are shown in an organized manner, improving the user's capacity to understand and examine pertinent data segments.

IV. Filtering

give the entries from <column> in <table> where <conditions>

The **filters** function manages user queries that include the 'where' clause, enabling accurate data retrieval according to defined conditions. Upon parsing the user's query to extract the desired table and conditions, the function proceeds to retrieve the existing data from the given table. After that, it uses the **checks** helper function to assess conditions for every row, selectively excluding entries that satisfy the stated requirements. The outcomes are exhibited in an organized manner via PrettyTable.

V. Insertion

```
include a new entry in <table> with <values>
```

The **insert** function plays a crucial role in adding new data into the dataset. The system analyzes the user's query to extract specific information about the new item, such as its values and the database it should be added to. The function retrieves the current data from the selected database, generates a new row with the given values, and smoothly incorporates this new entry. This user-friendly method of data insertion guarantees the system's versatility, allowing for new additions without compromising efficiency.

VI. Updating

```
edit <column> to <updated value> where <condition> in <table>
```

The **edit** function is essential for modifying data within the system. It efficiently manages user requests to modify specific data points inside a selected column of a provided table. The function carefully analyzes the user's query to extract significant information, including the target column, the updated value, the modification condition, and the table name. Afterwards, it traverses the current data in the designated table, identifying rows that meet the given requirement. After being found, the function carries out the specified modifications, guaranteeing that the dataset remains precise and represents real-world situations.

VII. Deletion

```
eliminate from <table> where <conditions>
```

An essential aspect of efficient data management is the capability to eliminate undesired items based on specified conditions, and the **eliminate** function achieves this task effortlessly. The function parses the user's query to obtain information regarding the target table and the criteria for removal. Following that, the program retrieves the current data from the designated table, eliminates entries that satisfy the stated criteria, and updates the table accordingly. This function guarantees that users can preserve data integrity by eliminating irrelevant or erroneous elements from their datasets.

VIII. Join

```
combine <table 1> and <table 2> using <common key> with <join
type>
```

The **combines** function enables the merging of data from two tables using a shared key and the provided type of join. The function parses the user's query to extract information about the tables, the common key, and the type of join to be executed (inner, left, right, or cross). The function performs the join operation by carefully manipulating data from both

tables and displays the initial 25 rows of the joined outcome. This feature greatly improves the system's flexibility, enabling users to combine different datasets for thorough analysis.

1. Inner Join: The Inner Join operation involves iterating over the rows of the second table (t2data). For every row, verifies the presence of the shared key (common) in the first table (t1data). If a match is found, the rows are merged, and the resulting data is added to the resultdata.

2. Left Join: Iterates through the rows of the first table (t1data). Each row is checked for a corresponding key in the second table (t2data). If the rows are located, they are merged; otherwise, a row with null values for the columns of the second table is appended.

3. Right Join: The right join is like the left join, except it iterates across the rows of the second table (t2data). The operation combines rows from the first table with matching rows or adds rows with null values if there is no match.

4. Cross Join: The cross join operation combines all rows from two tables, resulting in a Cartesian product. The rows from the first table are merged with the rows from the second table.

IX. Grouping

```
group <columns> from <table> with <other column> using
                <aggregation function>.
```

The **groupby** function executes user queries by implementing the 'group by' clause. The process retrieves the grouping columns, aggregation function, table name, and display columns. Once the aggregation function is validated, it proceeds to retrieve data from the designated database. The function then groups the data based on given columns and computes the aggregated results for each group using functions such as ***count, sum, min, max, or avg.*** The final aggregated results are exhibited utilizing PrettyTable, offering a clear depiction of grouped data.

X. Aggregation

```
compute <columns> from <table> using <function>
```

Data aggregation is essential, and the **aggregate** function enables users to extract significant insights from their information. The function collects information about the columns to aggregate, the destination table, and the desired aggregation function by parsing the user's query. After verifying the user-defined aggregation function, the function carefully examines the available data, carries out the aggregation, and displays the results in a well-organized fashion using PrettyTable. This feature improves the system's ability to carry out dynamic data analysis.

XI. Ordering

arrange the data <table> by <column> with <ascending/descending> order

Ordering and organizing data are fundamental activities in data processing. The **arrange** function simplifies this procedure by analyzing the user's query to extract information regarding the target table, the sorting column, and the preferred order (ascending or descending). Afterwards, the program retrieves the data from the designated table, performs the sorting operation according to the user's requirements, and displays the organized data using PrettyTable. This simple technique guarantees that users may rapidly get insights from the organized dataset.

b. Tech Stack

1. **Python:**

Chosen as the core programming language for its simplicity, readability, and versatility.

2. **Libraries:**

   a. **csv:**

   Role: Essential for handling CSV files, a common format for tabular data storage.
   Usage: Used for reading your RF signal dataset from CSV files and creating new CSV files when splitting the dataset into chunks.

   b. **os:**

   Role: Provides a way to interact with the operating system, enabling file and directory operations.
   Usage: Used for managing file paths, creating directories for storing tables, and checking the existence of files.

   c. **random:**

   Role: Facilitates random selection, useful when choosing a chunk file during the chunking process.
   Usage: Randomly selects a chunk file for processing in the **splitcsv** function.

3. **PrettyTable:**

Role: Enhances the presentation of query results by creating visually appealing ASCII tables.
Usage: Utilized to display tables with aligned columns, improving the readability of query outputs.

4. **Development Environments:**

   a. **Jupyter Notebook:**

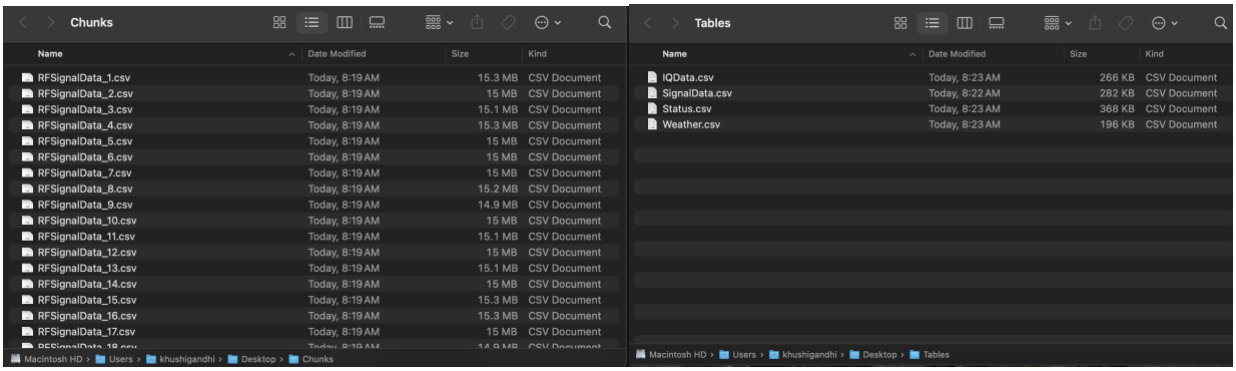
   Initial testing and function prototyping were carried out in Jupyter Notebooks. This interactive environment allowed for quick iterations and exploration of individual functions.

   b. **Visual Studio Code (VSCode):**

   The final implementation and integration of various functions into a cohesive system were done in Visual Studio Code. VSCode's powerful debugging tools and code navigation features enhanced the development process.

c. Implementation Screenshots

Chunks and Tables Folder



Creation of Table

```
Enter Query > produce table SignalData with columns: Timestamp datetime, Frequency int, SignalStrength int, Modulation string, Bandwidth int, DeviceType string, AntennaType string

+---------------+-----------+----------------+------------+-----------+------------+-----------------+
| Timestamp     | Frequency | SignalStrength | Modulation | Bandwidth | DeviceType | AntennaType     |
+---------------+-----------+----------------+------------+-----------+------------+-----------------+
| 5/26/23 23:46 | 160000000 | -70            | 8PSK       | 100000    | Halow-U    | Directional     |
| 5/26/23 23:47 | 160000000 | -83            | 8PSK       | 500000    | HackRF     | Yagi            |
| 5/26/23 23:47 | 160000000 | -19            | 8PSK       | 500000    | SteamDeck  | Directional     |
| 5/26/23 23:47 | 90000000  | -49            | 8PSK       | 50000     | HackRF     | Omnidirectional |
| 5/26/23 23:48 | 140000000 | -81            | 8PSK       | 10000     | SteamDeck  | Directional     |
| 5/26/23 23:48 | 140000000 | -89            | QAM        | 10000     | Halow-U    | Omnidirectional |
| 5/26/23 23:48 | 90000000  | -16            | 8PSK       | 500000    | Halow-U    | Directional     |
| 5/26/23 23:49 | 90000000  | -85            | QAM        | 500000    | SteamDeck  | Yagi            |
| 5/26/23 23:49 | 70000000  | -91            | FM         | 1000000   | Halow-U    | Dipole          |
| 5/26/23 23:49 | 70000000  | -45            | FM         | 10000     | SteamDeck  | Directional     |
| 5/26/23 23:50 | 90000000  | -62            | BPSK       | 100000    | HackRF     | Yagi            |
| 5/26/23 23:50 | 140000000 | -37            | AM         | 50000     | HackRF     | Yagi            |
| 5/26/23 23:50 | 90000000  | -12            | FM         | 1000000   | HackRF     | Omnidirectional |
| 5/26/23 23:51 | 100000000 | -95            | FM         | 100000    | HackRF     | Dipole          |
| 5/26/23 23:51 | 120000000 | -41            | BPSK       | 50000     | HackRF     | Dipole          |
| 5/26/23 23:51 | 120000000 | -30            | BPSK       | 1000000   | HackRF     | Dipole          |
| 5/26/23 23:52 | 100000000 | -47            | FM         | 200000    | HackRF     | Dipole          |
| 5/26/23 23:52 | 70000000  | -13            | QPSK       | 10000     | SteamDeck  | Omnidirectional |
| 5/26/23 23:52 | 160000000 | -13            | QAM        | 50000     | SteamDeck  | Dipole          |
| 5/26/23 23:53 | 70000000  | -45            | QPSK       | 100000    | Halow-U    | Omnidirectional |
| 5/26/23 23:53 | 140000000 | -10            | QAM        | 10000     | Halow-U    | Directional     |
| 5/26/23 23:53 | 70000000  | -41            | QAM        | 100000    | Halow-U    | Directional     |
| 5/26/23 23:54 | 120000000 | -52            | QAM        | 500000    | SteamDeck  | Dipole          |
| 5/26/23 23:54 | 70000000  | -64            | QAM        | 100000    | HackRF     | Omnidirectional |
| 5/26/23 23:54 | 70000000  | -40            | 8PSK       | 200000    | SteamDeck  | Omnidirectional |
| 5/26/23 23:55 | 90000000  | -10            | QAM        | 100000    | Halow-U    | Dipole          |
| 5/26/23 23:55 | 90000000  | -30            | 8PSK       | 200000    | HackRF     | Omnidirectional |
```

Projection

```
Enter Query > display Humidity, Precipitation from Weather

+----------+---------------+
| Humidity | Precipitation |
+----------+---------------+
|    65    |  17.49198967  |
|    69    |  19.48035363  |
|    36    |  16.78318309  |
|    59    |  15.17231916  |
|    41    |  40.41278137  |
|    26    |  9.617348167  |
|    45    |   9.39392851  |
|    68    |  38.11333959  |
|    20    |  6.577040607  |
|    38    |  36.16045177  |
|    67    |  32.89756346  |
|    72    |  45.17719805  |
|    47    |  36.59642451  |
|    35    |  37.85707573  |
|    65    |  5.498826058  |
|    22    |  23.26226922  |
|    45    |  12.15474558  |
|    54    |  36.14183332  |
|    24    |   9.31357838  |
|    63    |  29.19739767  |
|    21    |  40.47967063  |
|    22    |  20.68385919  |
|    30    |   41.7127145  |
|    31    |  32.40262725  |
|    54    |  28.51477591  |
|    54    |  12.42959805  |
|    28    |  45.01977101  |
```

```
Enter Query > display all from IQData

+---------------+------------+-----------------------------------------------------+
|   Timestamp   | DeviceType |                       IQData                        |
+---------------+------------+-----------------------------------------------------+
|  5/20/23 1:06 | SteamDeck  |                                                     |
|  5/20/23 1:07 |  Halow-U   |   "[(-0.0951651490830201-0.368455045093672555j)     |
|  5/20/23 1:07 |   HackRF   |   "[(0.024713308503058418-0.5060438700578673j)      |
|  5/20/23 1:07 |   HackRF   |    "[(0.7588967939794529-0.6076382598676919j)       |
|  5/20/23 1:08 |   HackRF   |   "[(-0.36561767029182657-0.6140299772408262j)      |
|  5/20/23 1:08 |   HackRF   |    "[(0.8639880849616306-0.4558481237149252j)       |
|  5/20/23 1:08 |  Halow-U   |    "[(0.7012262546411898+0.6716707476950519j)       |
|  5/20/23 1:09 | SteamDeck  |                                                     |
|  5/20/23 1:09 | SteamDeck  |                                                     |
|  5/20/23 1:09 |   HackRF   |   "[(-0.952407207384429-0.035982180463021374j)      |
|  5/20/23 1:10 |  Halow-U   |    "[(0.48136780789163813-0.9818921122567837j)      |
|  5/20/23 1:10 |   HackRF   |    "[(0.82644145156350354+0.7667812434115155j)      |
|  5/20/23 1:10 |   HackRF   |   "[(0.44211896738645806-0.8875369167213172j)       |
|  5/20/23 1:11 |   HackRF   |   "[(0.2246758455461486+0.014295904632607437j)      |
|  5/20/23 1:11 | SteamDeck  |                                                     |
|  5/20/23 1:11 | SteamDeck  |                                                     |
|  5/20/23 1:12 | SteamDeck  |                                                     |
|  5/20/23 1:12 |   HackRF   |   "[(-0.5257259899873883-0.5189842262687381j)       |
|  5/20/23 1:12 | SteamDeck  |                                                     |
|  5/20/23 1:13 |   HackRF   |    "[(0.8973452538401467-0.0880438536708541j)       |
|  5/20/23 1:13 |   HackRF   |  "[(0.2735066586231665-0.0018023306751260293j)      |
|  5/20/23 1:13 | SteamDeck  |                                                     |
|  5/20/23 1:14 | SteamDeck  |                                                     |
|  5/20/23 1:14 |  Halow-U   |    "[(0.6567689272772506-0.7452070569717708j)       |
|  5/20/23 1:14 |  Halow-U   |   "[(0.20733944341991895-0.8092903704754564j)       |
|  5/20/23 1:15 |   HackRF   |   "[(-0.5977819765667742-0.9294047733324997j)       |
|  5/20/23 1:15 |  Halow-U   |   "[(0.7388964835873797+0.37575184105032444j)       |
|  5/20/23 1:15 |  Halow-U   |   "[(0.33653321357401333+0.2031544137665790 6j)     |
|  5/20/23 1:16 | SteamDeck  |                                                     |
|  5/20/23 1:16 | SteamDeck  |                                                     |
|  5/20/23 1:16 |   HackRF   |   "[(-0.07891762910847921+0.568944888923592j)       |
```

Filtering

```
Enter Query > give the entries from Weather where WindSpeed > 10
Filtered Result from 'Weather':
```

Timestamp	Temperature	Humidity	WindSpeed	Precipitation	WeatherCondition
5/6/23 3:46	24	65	13	17.49198967	Sunny
5/6/23 3:47	27	59	14	15.17231916	Sunny
5/6/23 3:48	38	41	20	40.41278137	Cloudy
5/6/23 3:48	27	26	17	9.617348167	Sunny
5/6/23 3:49	21	68	14	38.11333959	Cloudy
5/6/23 3:49	38	20	12	6.577040607	Cloudy
5/6/23 3:50	31	67	19	32.89756346	Cloudy
5/6/23 3:50	20	72	11	45.17719805	Sunny
5/6/23 3:52	37	24	11	9.31357838	Cloudy
5/6/23 3:53	27	63	17	29.19739767	Rainy
5/6/23 3:53	30	22	18	20.68385919	Rainy
5/6/23 3:55	37	54	11	12.42959805	Cloudy
5/6/23 3:56	27	25	15	7.853521207	Rainy
5/6/23 3:56	33	46	17	38.38577017	Sunny
5/6/23 3:57	25	27	15	32.29918068	Sunny
5/6/23 3:57	28	66	19	21.90841552	Sunny
5/6/23 3:58	23	46	18	23.10137298	Sunny
5/6/23 4:00	35	61	20	15.17076057	Sunny
5/6/23 4:00	23	30	20	39.8390459	Sunny

Insertion

```
Enter Query > include a new entry in Weather with Timestamp == 6/5/23 09:00, Temperature == 35, Humidity == 40,
 WindSpeed == 23, Precipitation == 0, WeatherCondition == Sunny
New entry included in 'Weather': ['6/5/23 09:00', '35', '40', '23', '0', 'Sunny']
```

(after insertion)

5/7/23 7:31	36	37	19	36.44796491	Cloudy
5/7/23 7:32	39	25	13	36.1415901	Cloudy
5/7/23 7:32	20	26	0	14.98805	Sunny
5/7/23 7:32	31	52	20	25.87561978	Sunny
6/5/23 09:00	35	40	23	0	Sunny

Updation

```
Enter Query > edit InterferenceType to NA where InterferenceType == 'None' in Status
Data in 'InterferenceType' column updated to 'NA' where InterferenceType == 'None' in 'Status'.
Enter Query > display all from Status
```

Timestamp	InterferenceType	CPUUsage	MemoryUsage	WiFiStrength	DiskUsage	SystemLoad	DeviceStatus
5/26/23 23:46	Adjacent-channel	0	29.5	-3	18.5	4.607421875	Transmitting beacon signal
5/26/23 23:47	NA	100	29.5	-64	18.5	4.607421875	Streaming I/Q data
5/26/23 23:47	Intermodulation	0	29.5	-6	18.5	4.607421875	Running game
5/26/23 23:47	Co-channel	0	29.5	-4	18.5	4.607421875	Streaming I/Q data
5/26/23 23:48	Adjacent-channel	0	29.5	-32	18.5	4.607421875	Running game
5/26/23 23:48	NA	0	29.5	-76	18.5	4.607421875	Transmitting beacon signal
5/26/23 23:48	Adjacent-channel	0	29.5	-43	18.5	4.607421875	Transmitting beacon signal
5/26/23 23:49	Adjacent-channel	50	29.5	-14	18.5	4.607421875	Running game
5/26/23 23:49	Adjacent-channel	0	29.5	-83	18.5	4.607421875	Transmitting beacon signal
5/26/23 23:49	Adjacent-channel	0	29.5	-34	18.5	4.607421875	Running game
5/26/23 23:50	NA	0	29.5	-92	18.5	4.607421875	Streaming I/Q data

Deletion

```
Enter Query > eliminate from Weather where Precipitation == '0'
Entries eliminated from 'Weather' where Precipitation == '0'.
Enter Query >
```

(after deletion)

5/7/23 7:30	23	50	5	28.95506712	Cloudy
5/7/23 7:31	20	76	10	42.03381329	Sunny
5/7/23 7:31	24	28	11	39.33905825	Cloudy
5/7/23 7:31	36	37	19	36.44796491	Cloudy
5/7/23 7:32	39	25	13	36.1415901	Cloudy
5/7/23 7:32	20	26	0	14.98805	Sunny
5/7/23 7:32	31	52	20	25.87561978	Sunny

Grouping

```
Enter Query > group WeatherCondition from Weather with Humidity using sum
Aggregated Result from 'Weather':
+------------------+-----------------+
| WeatherCondition | Humidity (Sum) |
+------------------+-----------------+
| Sunny            | 85926.0        |
| Cloudy           | 83661.0        |
| Rainy            | 79333.0        |
+------------------+-----------------+
```

Aggregation

```
Enter Query > compute Temperature, Humidity from Weather using sum
Aggregated Result from 'Weather':
+-------------+--------------------+
| Column      | Aggregation Result |
+-------------+--------------------+
| Temperature | 149663.0           |
| Humidity    | 248920.0           |
+-------------+--------------------+
Enter Query > compute Temperature, Humidity from Weather using avg
Aggregated Result from 'Weather':
+-------------+--------------------+
| Column      | Aggregation Result |
+-------------+--------------------+
| Temperature | 29.93858771754351  |
| Humidity    | 49.79395879175835  |
+-------------+--------------------+
Enter Query > compute Temperature, Humidity from Weather using count
Aggregated Result from 'Weather':
+-------------+--------------------+
| Column      | Aggregation Result |
+-------------+--------------------+
| Temperature | 4999               |
| Humidity    | 4999               |
+-------------+--------------------+
Enter Query > compute Temperature, Humidity from Weather using max
Aggregated Result from 'Weather':
+-------------+--------------------+
| Column      | Aggregation Result |
+-------------+--------------------+
| Temperature | 40.0               |
| Humidity    | 80.0               |
+-------------+--------------------+
Enter Query > compute Temperature, Humidity from Weather using min
Aggregated Result from 'Weather':
+-------------+--------------------+
| Column      | Aggregation Result |
+-------------+--------------------+
| Temperature | 20.0               |
| Humidity    | 20.0               |
+-------------+--------------------+
```

Ordering

```
Enter Query > arrange the data in IQData by DeviceType with ascending order
Arranged Data in 'IQData' by 'DeviceType' with 'ascending' order:
+---------------+------------+-------------------------------------------------------+
| Timestamp     | DeviceType | IQData                                                |
+---------------+------------+-------------------------------------------------------+
| 5/20/23 1:07  | HackRF     | "[(0.024713308503058418-0.5060438700578673j)          |
| 5/20/23 1:07  | HackRF     | "[(0.7588967939794529-0.6076382598676919j)            |
| 5/20/23 1:08  | HackRF     | "[(-0.36561767029182657-0.6140297972408262j)          |
| 5/20/23 1:08  | HackRF     | "[(0.8639880849616306-0.4558481237149252j)            |
| 5/20/23 1:09  | HackRF     | "[(-0.952407207384429-0.035982180463021374j)          |
| 5/20/23 1:10  | HackRF     | "[(0.8264414515635035+0.7667812434115155j)            |
| 5/20/23 1:10  | HackRF     | "[(0.44211896738645806-0.8875369167213172j)           |
| 5/20/23 1:11  | HackRF     | "[(0.2246758455461486+0.01429590463260743 7j)         |
| 5/20/23 1:12  | HackRF     | "[(-0.5257259899873883-0.5189842262687381j)           |
| 5/20/23 1:13  | HackRF     | "[(0.8973452538401467-0.0880438536708541j)            |
| 5/20/23 1:13  | HackRF     | "[(0.2735066586231665-0.0018023306751260293j)         |
| 5/20/23 1:15  | HackRF     | "[(-0.5977819765667742-0.9294047733324997j)           |
| 5/20/23 1:16  | HackRF     | "[(-0.07891762910847921+0.568944888923592j)           |
| 5/20/23 1:17  | HackRF     | "[(-0.30831870139986806-0.695311214998759j)           |
| 5/20/23 1:18  | HackRF     | "[(-0.24326060898693913-0.4636461844672728j)          |
| 5/20/23 1:18  | HackRF     | "[(-0.8288994739083959+0.4252375199632821j)           |
| 5/20/23 1:18  | HackRF     | "[(0.7823923097554+0.2992792475593298j)               |
| 5/20/23 1:19  | HackRF     | "[(-0.3133498876487495-0.4138089594508694j)           |
| 5/20/23 1:19  | HackRF     | "[(0.6973615233224724+0.5546345668773154j)            |
| 5/20/23 1:19  | HackRF     | "[(-0.3177923588748175-0.24555340283065652j)          |
```

Join

(Cross Join)

```
Enter Query > combine SignalData and IQData using DeviceType with cross

First 25 Rows of Combined Result of 'SignalData' and 'IQData' using 'DeviceType' with 'cross' join:
+-------------+-----------+----------------+------------+-----------+------------+-------------+-----------------------------------------------+
| Timestamp   | Frequency | SignalStrength | Modulation | Bandwidth | DeviceType | AntennaType | IQData                                        |
+-------------+-----------+----------------+------------+-----------+------------+-------------+-----------------------------------------------+
| 5/20/23 1:06 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:07 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole      | "[(-0.0951651490830201-0.36845504509367255j)  |
| 5/20/23 1:07 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(0.024713308503058418-0.5060438700578673j)  |
| 5/20/23 1:07 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(0.7588967939794529-0.6076382598676919j)    |
| 5/20/23 1:08 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(-0.36561767029182657-0.6140297972408262j)  |
| 5/20/23 1:08 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(0.8639880849616306-0.4558481237149252j)    |
| 5/20/23 1:08 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole      | "[(0.7012262546411898+0.6716707476950519j)    |
| 5/20/23 1:09 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:09 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:09 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(-0.952407207384429-0.035982180463021374j)  |
| 5/20/23 1:10 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole      | "[(0.48136780789163813-0.981892112256783j)    |
| 5/20/23 1:10 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(0.8264414515635035+0.7667812434115155j)    |
| 5/20/23 1:10 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(0.44211896738645806-0.8875369167213172j)   |
| 5/20/23 1:11 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(0.2246758455461486+0.014295904632607437j)  |
| 5/20/23 1:11 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:11 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:12 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:12 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(-0.5257259899873883-0.5189842262687381j)   |
| 5/20/23 1:12 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:13 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(0.8973452538401467-0.0880438536708541j)    |
| 5/20/23 1:13 | 100000000 | -64          | 8PSK       | 1000000   | HackRF     | Dipole      | "[(0.2735066586231665-0.0018023306751260293j) |
| 5/20/23 1:13 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:14 | 100000000 | -64          | 8PSK       | 1000000   | SteamDeck  | Dipole      |                                               |
| 5/20/23 1:14 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole      | "[(0.6567689272772506-0.7452070569717708j)    |
| 5/20/23 1:14 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole      | "[(0.20733944341991895-0.8092903704754564j)   |
+-------------+-----------+----------------+------------+-----------+------------+-------------+-----------------------------------------------+
```

(Right Join)

```
Enter Query > combine SignalData and IQData using DeviceType with right

First 25 Rows of Combined Result of 'SignalData' and 'IQData' using 'DeviceType' with 'right' join:
+-------------+-----------+----------------+------------+-----------+------------+-----------------+-----------------------------------------------+
| Timestamp   | Frequency | SignalStrength | Modulation | Bandwidth | DeviceType | AntennaType     | IQData                                        |
+-------------+-----------+----------------+------------+-----------+------------+-----------------+-----------------------------------------------+
| 5/20/23 1:06 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:07 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole          | "[(-0.0951651490830201-0.36845504509367255j)  |
| 5/20/23 1:07 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.024713308503058418-0.5060438700578673j)  |
| 5/20/23 1:07 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.7588967939794529-0.6076382598676919j)    |
| 5/20/23 1:08 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(-0.36561767029182657-0.6140297972408262j)  |
| 5/20/23 1:08 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.8639880849616306-0.4558481237149252j)    |
| 5/20/23 1:08 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole          | "[(0.7012262546411898+0.6716707476950519j)    |
| 5/20/23 1:09 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:09 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:09 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(-0.952407207384429-0.035982180463021374j)  |
| 5/20/23 1:10 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole          | "[(0.48136780789163813-0.981892112256783j)    |
| 5/20/23 1:10 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.8264414515635035+0.7667812434115155j)    |
| 5/20/23 1:10 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.44211896738645806-0.8875369167213172j)   |
| 5/20/23 1:11 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.2246758455461486+0.014295904632607437j)  |
| 5/20/23 1:11 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:11 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:12 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:12 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(-0.5257259899873883-0.5189842262687381j)   |
| 5/20/23 1:12 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:13 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.8973452538401467-0.0880438536708541j)    |
| 5/20/23 1:13 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.2735066586231665-0.0018023306751260293j) |
| 5/20/23 1:13 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:14 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
| 5/20/23 1:14 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole          | "[(0.6567689272772506-0.7452070569717708j)    |
| 5/20/23 1:14 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole          | "[(0.20733944341991895-0.8092903704754564j)   |
+-------------+-----------+----------------+------------+-----------+------------+-----------------+-----------------------------------------------+
```

(Left Join)

```
Enter Query > combine SignalData and IQData using DeviceType with left

First 25 Rows of Combined Result of 'SignalData' and 'IQData' using 'DeviceType' with 'left' join:
+-------------+-----------+----------------+------------+-----------+------------+-----------------+-----------------------------------------------+
| Timestamp   | Frequency | SignalStrength | Modulation | Bandwidth | DeviceType | AntennaType     | IQData                                        |
+-------------+-----------+----------------+------------+-----------+------------+-----------------+-----------------------------------------------+
| 5/20/23 1:07 | 100000000 | -64          | 8PSK       | 1000000   | Halow-U    | Dipole          | "[(-0.0951651490830201-0.36845504509367255j)  |
| 5/20/23 1:07 | 100000000 | -53          | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.024713308503058418-0.5060438700578673j)  |
| 5/20/23 1:06 | 90000000  | -78          | BPSK       | 10000     | SteamDeck  | Directional     |                                               |
+-------------+-----------+----------------+------------+-----------+------------+-----------------+-----------------------------------------------+
```

(Inner Join)

```
Enter Query > combine SignalData and IQData using DeviceType with inner

First 25 Rows of Combined Result of 'SignalData' and 'IQData' using 'DeviceType' with 'inner' join:
+--------------+-----------+----------------+------------+-----------+------------+----------------+-----------------------------------------------+
| Timestamp    | Frequency | SignalStrength | Modulation | Bandwidth | DeviceType | AntennaType    | IQData                                        |
+--------------+-----------+----------------+------------+-----------+------------+----------------+-----------------------------------------------+
| 5/20/23 1:06 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:07 | 100000000 | -64            | 8PSK       | 1000000   | Halow-U    | Dipole         | "[(-0.0951651490830201-0.36845504509367255j)  |
| 5/20/23 1:07 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.024713308503058418-0.5060438700578673j)  |
| 5/20/23 1:07 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.7588967939794529-0.6076382598676919j)    |
| 5/20/23 1:08 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(-0.36561767029182657-0.6140297972408262j)  |
| 5/20/23 1:08 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.8639880849616306-0.4558481237149252j)    |
| 5/20/23 1:08 | 100000000 | -64            | 8PSK       | 1000000   | Halow-U    | Dipole         | "[(0.7012262546411898+0.6716707476950519j)    |
| 5/20/23 1:09 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:09 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:09 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(-0.952407207384429-0.035982180463021374j)  |
| 5/20/23 1:10 | 100000000 | -64            | 8PSK       | 1000000   | Halow-U    | Dipole         | "[(0.48136780789163813-0.9818921122567 83j)   |
| 5/20/23 1:10 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.8264414515635035+0.7667812434115155j)    |
| 5/20/23 1:10 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.44211896738645806-0.8875369167213172j)   |
| 5/20/23 1:11 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.2246758455461486+0.014295904632607437j)  |
| 5/20/23 1:11 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:11 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:12 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:12 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(-0.5257259899873883-0.5189842262687381j)   |
| 5/20/23 1:12 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:13 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.8973452538401467-0.0880438536708541j)    |
| 5/20/23 1:13 | 100000000 | -53            | QPSK       | 100000    | HackRF     | Omnidirectional | "[(0.2735066586231665-0.0018023306751260293j) |
| 5/20/23 1:13 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:14 | 90000000  | -78            | BPSK       | 10000     | SteamDeck  | Directional    |                                               |
| 5/20/23 1:14 | 100000000 | -64            | 8PSK       | 1000000   | Halow-U    | Dipole         | "[(0.6567689272772506-0.7452070569717708j)    |
| 5/20/23 1:14 | 100000000 | -64            | 8PSK       | 1000000   | Halow-U    | Dipole         | "[(0.20733944341991895-0.8092903704754564j)   |
+--------------+-----------+----------------+------------+-----------+------------+----------------+-----------------------------------------------+
```

CLI

```
(base) khushigandhi@Khushis-Air Desktop % python Final.py
**********************************************************************************************
                        Welcome to the Command Line Prompt
**********************************************************************************************
Enter Query >
```

V. Learning Outcome

- **Designing a DB System:** This project offered practical exposure to creating a personalized database system, comprehending the fundamentals of relational databases, and executing tasks such as querying, updating, and aggregating data.

- **Query Language Design:** The creation of a specialized query language required careful consideration of grammar, interpreting user input, and converting queries into executable operations on the dataset. This technique improved language design skills and fostered logical thinking.

- **Improved Python Proficiency:** Working extensively with Python, especially modules like PrettyTable for result display and CSV for data handling, improved my fluency with the language. The project additionally encompassed tasks related to file input/output, data manipulation, and algorithm implementation.

- **Memory Management:** Employing chunking to manage huge datasets and prevent the need to load the full database into memory enhanced comprehension of memory management in practical situations.

a. Challenges Faced

- **Difficulties with ANTLR:** During the project, the incorporation of ANTLR presented significant difficulties, especially in debugging the grammar file after the parser and lexer files were generated. The complex nature of addressing problems inside the code created by ANTLR required a substantial amount of time and hindered the pace of the project. To optimize and improve effectiveness, a deliberate choice was made to discontinue the utilization of ANTLR. The transition to Python not only accelerated the process of identifying and resolving errors, but also enhanced the entire implementation experience.

- **Filtering:** The implementation of filtering, particularly the management of intricate situations, presented difficulties. Converting user-defined conditions into code that can be executed required meticulous attention to syntax and logic.

- **Grouping:** The task of designing and implementing the 'group by' functionality posed a challenge due to the complex process of grouping data according to defined columns. Incorporating several scenarios into the aggregating process increased the complexity.

- **Gap Between Concept and Code:** Although possessing a thorough comprehension of logic, the process of converting thoughts into operational code presented unexpected challenges. The process of transforming a conceptualized feature into functional code demanded iterative problem-solving and debugging.

VI. Conclusion

The objective of this project is to develop a customized database system for a sizable dataset, drawing inspiration from MySQL. The system enables users to efficiently engage with the data through a range of functionalities, including projection, filtering, joining, grouping, aggregating, ordering, and CRUD activities. The implementation of Python instead of ANTLR streamlined the development process and enhanced efficiency. The system manages massive data sets by manually dividing them into memory-efficient chunks. Although there were difficulties in implementing filtering and grouping functions, the final product successfully provides a functioning and user-friendly database system that can be easily adjusted to accommodate various datasets. The technology stack, predominantly utilizing Python, guarantees ease of use, while the utilization of Jupyter Notebook for testing and VSCode for the final implementation showcases a thorough development methodology. The learning outcomes encompass proficiency in technical skills and adeptness in problem-solving, with a particular focus on practical applications. The iterative nature of development is highlighted by the challenges encountered in ANTLR, filtering, and grouping. This project exemplifies the tedious yet gratifying process of constructing a database system, tackling practical obstacles, and acquiring a deep understanding of the intricacies of data management.

VII. Future Scope

- **Enhanced Grouping Function:** To enhance the capacity to group data in more advanced manners, I propose incorporating the combination of grouping and ordering to yield greater insights. In addition, I would like to experiment with incorporating grouping by use joins to combine data from other tables.

- **Enhanced Filtering Function:** To incorporate an improved filtering mechanism that can effectively manage intricate criteria, particularly in situations that involve data from numerous tables. Users will have the capability to apply filters to entries and join tables.

- **User-Friendly Interface:** As this project has a CLI implemented, I want to try and shift from a text-based command system to a more user-friendly app or visual interface. Creating a simple and visually appealing interface would make it easier for users to interact with the system, accommodating a wider audience.

All these future scope goals lead towards improving user experience and a promise to achieving a smoother system which can handle complexities.

VIII. References

[1] https://www.kaggle.com/datasets/suraj520/rf-signal-data

[2] https://www.w3schools.com/mysql/mysql_sql.asp