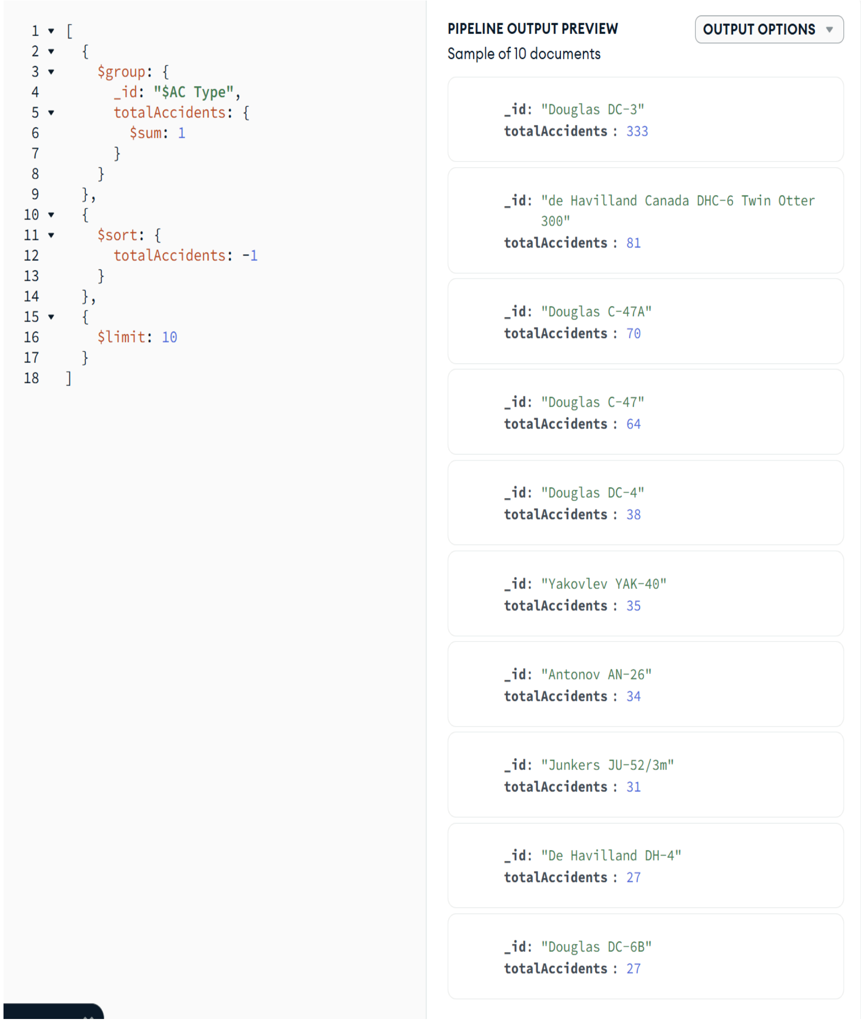
Part 1: Typical Research Ideas:

Aircraft and Operator Safety Analysis

- **Number 1:**
    - **Question:** Most Frequent Crash Type
    - **Fields Used:** AC Type
    - **Goal:** What are the top 10 aircraft models involved in the highest number of accidents?

Prompt: Provide the top 10 aircraft models involved in the highest number of accidents?

```
[
 {
  $group: {
   _id: "$AC Type",
   totalAccidents: {
    $sum: 1
   }
  }
 },
 {
  $sort: {
   totalAccidents: -1
  }
 },
 {
  $limit: 10
 }
]
```

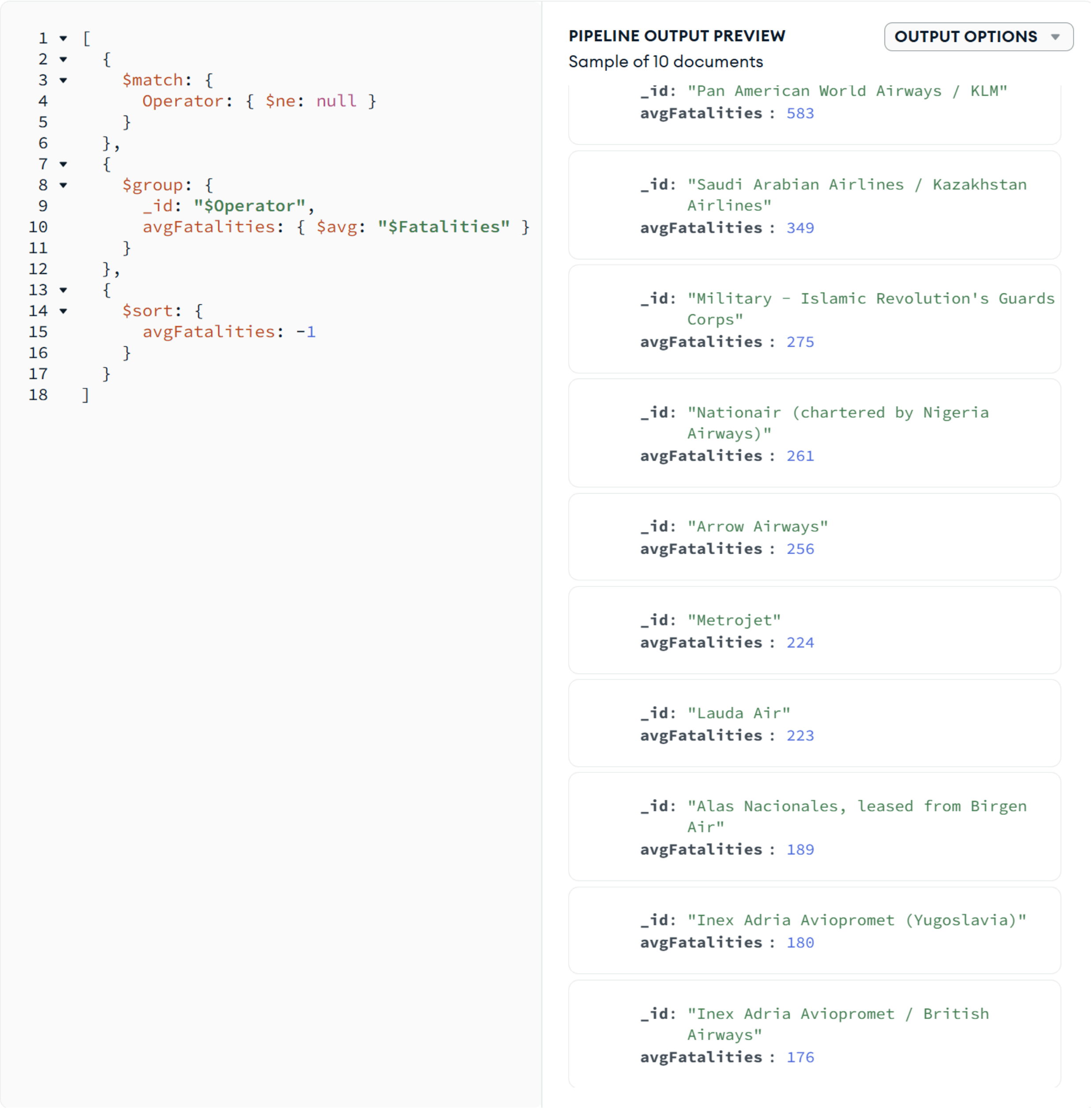


```
[
  {
    $group: {
      _id: "$AC Type",
      totalAccidents: {
        $sum: 1
      }
    }
  },
  {
    $sort: {
      totalAccidents: -1
    }
  },
  {
    $limit: 10
  }
]
```

```
_id: "Douglas DC-3"
totalAccidents : 333

_id: "de Havilland Canada DHC-6 Twin Otter
     300"
totalAccidents : 81

_id: "Douglas C-47A"
totalAccidents : 70

_id: "Douglas C-47"
totalAccidents : 64

_id: "Douglas DC-4"
totalAccidents : 38

_id: "Yakovlev YAK-40"
totalAccidents : 35

_id: "Antonov AN-26"
totalAccidents : 34

_id: "Junkers JU-52/3m"
totalAccidents : 31

_id: "De Havilland DH-4"
totalAccidents : 27

_id: "Douglas DC-6B"
totalAccidents : 27
```

- **Number 2**
  - **Question:** Worst Fatalities per Crash
  - **Fields Used:** Operator, Fatalities
  - **Goal:** Which commercial operators (airlines) have the highest average number of fatalities per crash?

Prompt: Which commercial operators (airlines) have the highest average number of fatalities per crash?



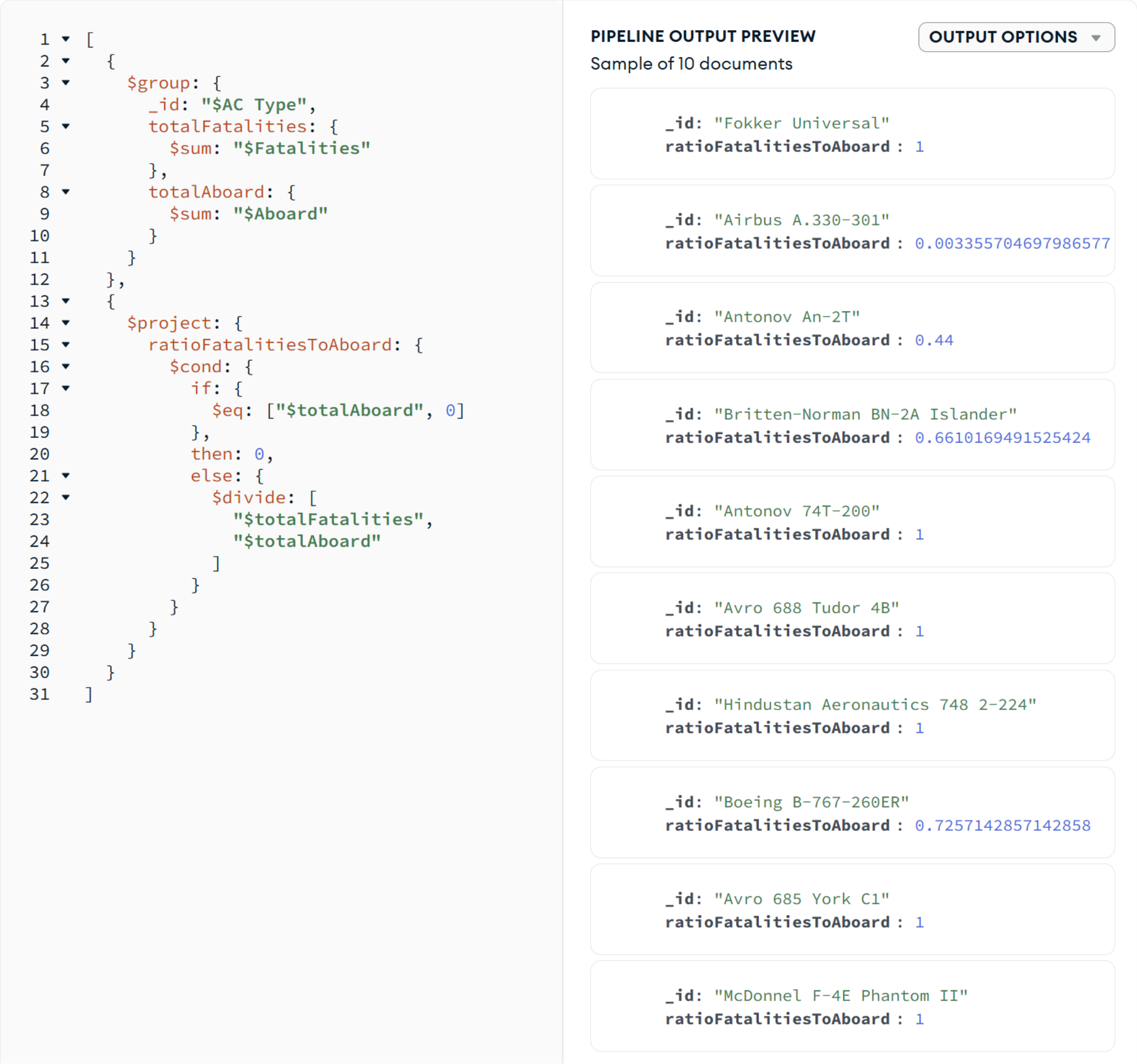
```
[
  {
    $match: {
      Operator: { $ne: null }
    }
  },
  {
    $group: {
      _id: "$Operator",
      avgFatalities: { $avg: "$Fatalities" }
    }
  },
  {
    $sort: {
      avgFatalities: -1
    }
  }
]
```

Sample of 10 documents

```
_id: "Pan American World Airways / KLM"
avgFatalities : 583

_id: "Saudi Arabian Airlines / Kazakhstan
     Airlines"
avgFatalities : 349

_id: "Military - Islamic Revolution's Guards
     Corps"
avgFatalities : 275

_id: "Nationair (chartered by Nigeria
     Airways)"
avgFatalities : 261

_id: "Arrow Airways"
avgFatalities : 256

_id: "Metrojet"
avgFatalities : 224

_id: "Lauda Air"
avgFatalities : 223

_id: "Alas Nacionales, leased from Birgen
     Air"
avgFatalities : 189

_id: "Inex Adria Aviopromet (Yugoslavia)"
avgFatalities : 180

_id: "Inex Adria Aviopromet / British
     Airways"
avgFatalities : 176
```

- **Number 3**
  - **Question:** Highest Risk Ratio
  - **Fields Used:** Fatalities, Aboard
  - **Goal:** For a specific aircraft type (e.g. Boeing 747), what is the ratio of fatalities to persons aboard across all incidents?

Prompt: Calculate the ratio of fatalities to people aboard across all incidents, grouped by each specific type of aircraft?



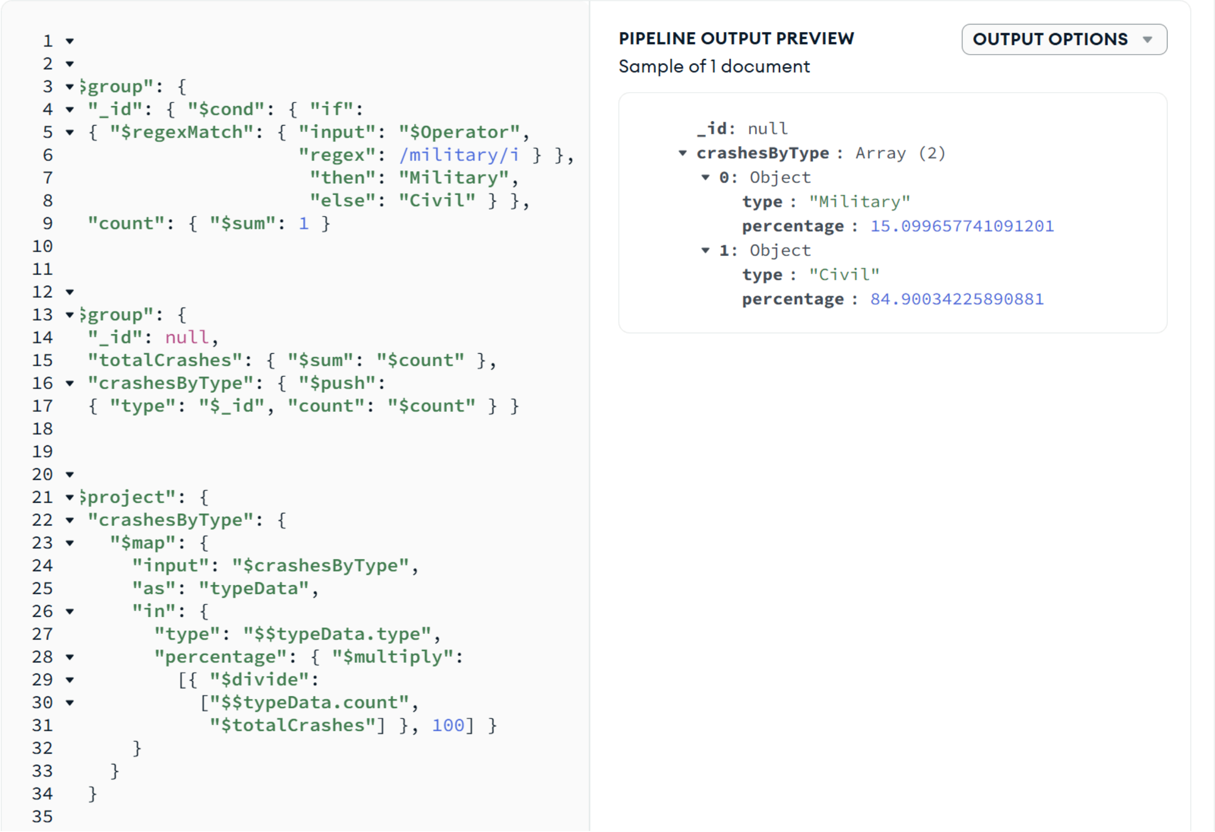
```
[
  {
    $group: {
      _id: "$AC Type",
      totalFatalities: {
        $sum: "$Fatalities"
      },
      totalAboard: {
        $sum: "$Aboard"
      }
    }
  },
  {
    $project: {
      ratioFatalitiesToAboard: {
        $cond: {
          if: {
            $eq: ["$totalAboard", 0]
          },
          then: 0,
          else: {
            $divide: [
              "$totalFatalities",
              "$totalAboard"
            ]
          }
        }
      }
    }
  }
]
```

Sample of 10 documents

```
_id: "Fokker Universal"
ratioFatalitiesToAboard : 1

_id: "Airbus A.330-301"
ratioFatalitiesToAboard : 0.003355704697986577

_id: "Antonov An-2T"
ratioFatalitiesToAboard : 0.44

_id: "Britten-Norman BN-2A Islander"
ratioFatalitiesToAboard : 0.6610169491525424

_id: "Antonov 74T-200"
ratioFatalitiesToAboard : 1

_id: "Avro 688 Tudor 4B"
ratioFatalitiesToAboard : 1

_id: "Hindustan Aeronautics 748 2-224"
ratioFatalitiesToAboard : 1

_id: "Boeing B-767-260ER"
ratioFatalitiesToAboard : 0.7257142857142858

_id: "Avro 685 York C1"
ratioFatalitiesToAboard : 1

_id: "McDonnel F-4E Phantom II"
ratioFatalitiesToAboard : 1
```

- **Number 4**
    - **Question:** Military vs Civil Safety
    - **Fields Used:** Operator
    - **Goal:** What percentage of total crashes involve military operators versus civil/commercial operators?

Prompt: What percentage of total crashes involve military operators versus civil/commercial operators?



```


$group": {
 "_id": { "$cond": { "if":
 { "$regexMatch": { "input": "$Operator",
                    "regex": /military/i } },
                     "then": "Military",
                     "else": "Civil" } },
 "count": { "$sum": 1 }



$group": {
 "_id": null,
 "totalCrashes": { "$sum": "$count" },
 "crashesByType": { "$push":
 { "type": "$_id", "count": "$count" } }



$project": {
 "crashesByType": {
   "$map": {
     "input": "$crashesByType",
     "as": "typeData",
     "in": {
       "type": "$$typeData.type",
       "percentage": { "$multiply":
         [{ "$divide":
           ["$$typeData.count",
            "$totalCrashes"] }, 100] }
     }
   }
 }

```

```
_id: null
▾ crashesByType : Array (2)
  ▾ 0: Object
      type : "Military"
      percentage : 15.099657741091201
  ▾ 1: Object
      type : "Civil"
      percentage : 84.90034225890881
```

Aircraft and Operator Safety questions reflection: It was quite easy to utilize the LLM when answering these questions. I did not run into any issues or errors. I did not tweak anything when importing the data, and MongoDB was smart enough to import all of the needed fields as the correct data type (no integers imported as strings, for example).

Temporal Pattern Identification:

- **Number 1**
  - **Question:** Fatalities Trend
  - **Fields Used:** Date, Fatalities
  - **Goal:** How has the total number of fatalities per decade changed since the 1950s (looking for general improvement/deterioration)?

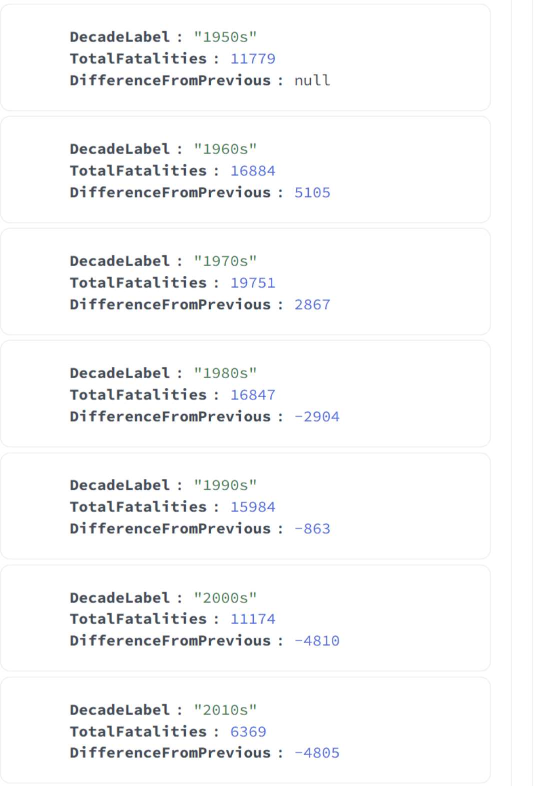
Prompt: Generate a MongoDB aggregation pipeline that starts with accidents from 1950 onward. The field "Date" is stored as a string in the format "MM/DD/YYYY". Extract the year from the "Date" string with $substr and convert it to an integer, for example year. Filter out any records where year is less than 1950. Compute the decade start as year minus (year modulo 10) and store it in a field like decadeStart. Group by decadeStart and sum the "Fatalities" field into totalFatalities. Sort the grouped results by decadeStart in ascending order. Group again into a single document that contains two arrays: one array of the sorted decadeStart values and one array of the corresponding totalFatalities values, using $group with $push. In a $project stage, use build an array of documents where each element contains: the decade start, the total fatalities for that decade, and the difference from the previous decade. For index 0, set the difference to null or 0. For later indices, subtract the previous decade total from the current decade total. $unwind that array so that you return one document per decade. Output a field DecadeLabel like "1950s", along with TotalFatalities and DifferenceFromPrevious, and exclude _id.

Query:

```
[
  {
    "$addFields": {
      "year": {
        "$toInt": {
          "$substr": ["$Date", 6, 4]
        }
      }
    }
  },
  {
    "$match": {
      "year": {
        "$gte": 1950
      }
    }
  },
  {
    "$addFields": {
      "decadeStart": {
        "$subtract": ["$year", { "$mod": ["$year", 10] }]
      }
    }
  },
  {
    "$group": {
      "_id": "$decadeStart",
      "totalFatalities": {
        "$sum": "$Fatalities"
      }
    }
  },
  {
    "$sort": {
      "_id": 1
    }
  },
  {
    "$group": {
      "_id": null,
      "decadeStarts": {
        "$push": "$_id"
      },
      "totalFatalitiesList": {
        "$push": "$totalFatalities"
      }
    }
  },
  {
    "$project": {
      "combinedArray": {
        "$map": {
          "input": {
            "$range": [0, { "$size": "$decadeStarts" }]
          },
          "as": "index",
          "in": {
            "decadeStart": { "$arrayElemAt": ["$decadeStarts", "$$index"] },
            "totalFatalities": { "$arrayElemAt": ["$totalFatalitiesList", "$$index"] },
            "difference": {
              "$cond": {
                "if": { "$eq": ["$$index", 0] },
                "then": null,
                "else": {
```

```
         "$subtract": [
          { "$arrayElemAt": ["$totalFatalitiesList", "$$index"] },
          { "$arrayElemAt": ["$totalFatalitiesList", { "$subtract": ["$$index", 1] }] }
         ]
        }
       }
      }
     }
    }
   }
  }
 },
 {
  "$unwind": "$combinedArray"
 },
 {
  "$project": {
   "DecadeLabel": {
    "$concat": [
     { "$toString": "$combinedArray.decadeStart" },
     "s"
    ]
   },
   "TotalFatalities": "$combinedArray.totalFatalities",
   "DifferenceFromPrevious": "$combinedArray.difference",
   "_id": 0
  }
 }
]
```

Results:



```
DecadeLabel : "1950s"
TotalFatalities : 11779
DifferenceFromPrevious : null

DecadeLabel : "1960s"
TotalFatalities : 16884
DifferenceFromPrevious : 5105

DecadeLabel : "1970s"
TotalFatalities : 19751
DifferenceFromPrevious : 2867

DecadeLabel : "1980s"
TotalFatalities : 16847
DifferenceFromPrevious : -2904

DecadeLabel : "1990s"
TotalFatalities : 15984
DifferenceFromPrevious : -863

DecadeLabel : "2000s"
TotalFatalities : 11174
DifferenceFromPrevious : -4810

DecadeLabel : "2010s"
TotalFatalities : 6369
DifferenceFromPrevious : -4805
```

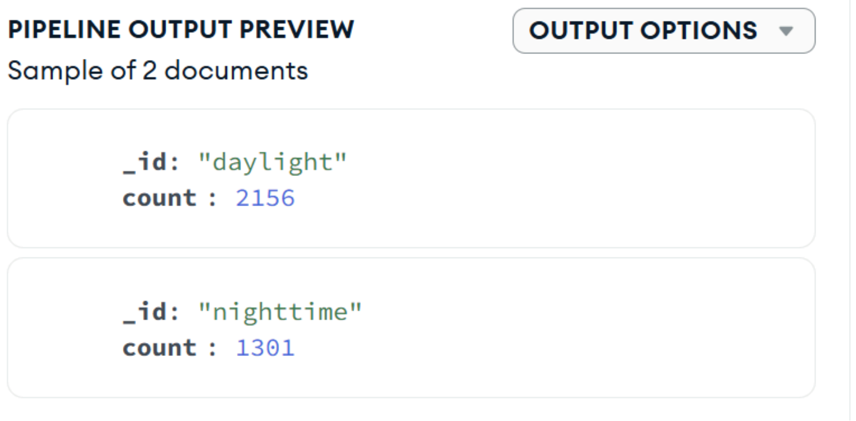
- **Number 2**
    - **Question:** Risk by Time of Day
    - **Fields Used:** Time
    - **Goal**: Are accidents significantly more common during daylight (e.g. 06:00 to 18:00) or nighttime hours?

Prompt: Are accidents significantly more common during daylight (6am to 6pm) or nighttime hours?

Query:

```
[
  {
    $addFields: {
      time_parts: { $split: ["$Time", ":"] }
    }
  },
  {
    $project: {
      hour: { $arrayElemAt: ["$time_parts", 0] },
      Date: 1
    }
  },
  {
    $match: {
      hour: { $gte: "06", $lt: "18" }
    }
  },
  {
    $group: {
      _id: "daylight",
      count: { $sum: 1 }
    }
  },
  {
    $unionWith: {
      coll: "airplane_crashes",
      pipeline: [
        {
          $addFields: {
            time_parts: { $split: ["$Time", ":"] }
          }
        },
        {
          $project: {
            hour: { $arrayElemAt: ["$time_parts", 0] },
            Date: 1
          }
        },
        {
          $match: {
            $or: [
              { hour: { $lt: "06" } },
              { hour: { $gte: "18" } }
            ]
          }
        },
        {
          $group: {
            _id: "nighttime",
            count: { $sum: 1 }
          }
        }
      ]
    }
  }
]
```

Result:



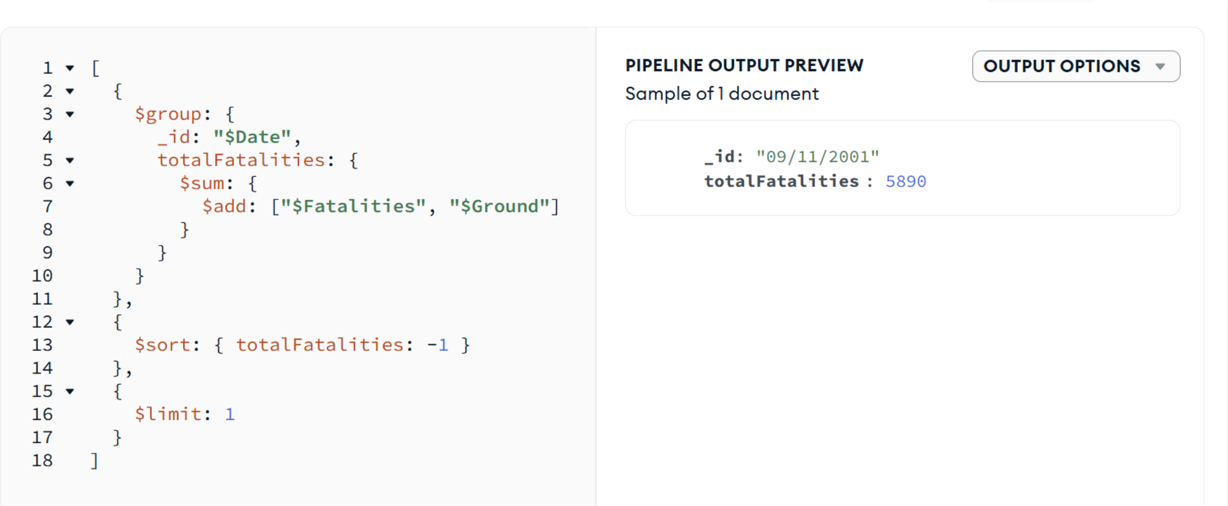
Sample of 2 documents

```
_id: "daylight"
count : 2156
```

```
_id: "nighttime"
count : 1301
```

- **Number 3**
    - **Question:** Worst Single Accident
    - **Fields Used:** Date, Fatalities, Ground
    - **Goal:** Which single date has the highest combined total of onboard and ground fatalities?

Prompt: Which single date has the highest combined total of onboard and ground fatalities?



```
[
  {
    $group: {
      _id: "$Date",
      totalFatalities: {
        $sum: {
          $add: ["$Fatalities", "$Ground"]
        }
      }
    }
  },
  {
    $sort: { totalFatalities: -1 }
  },
  {
    $limit: 1
  }
]
```

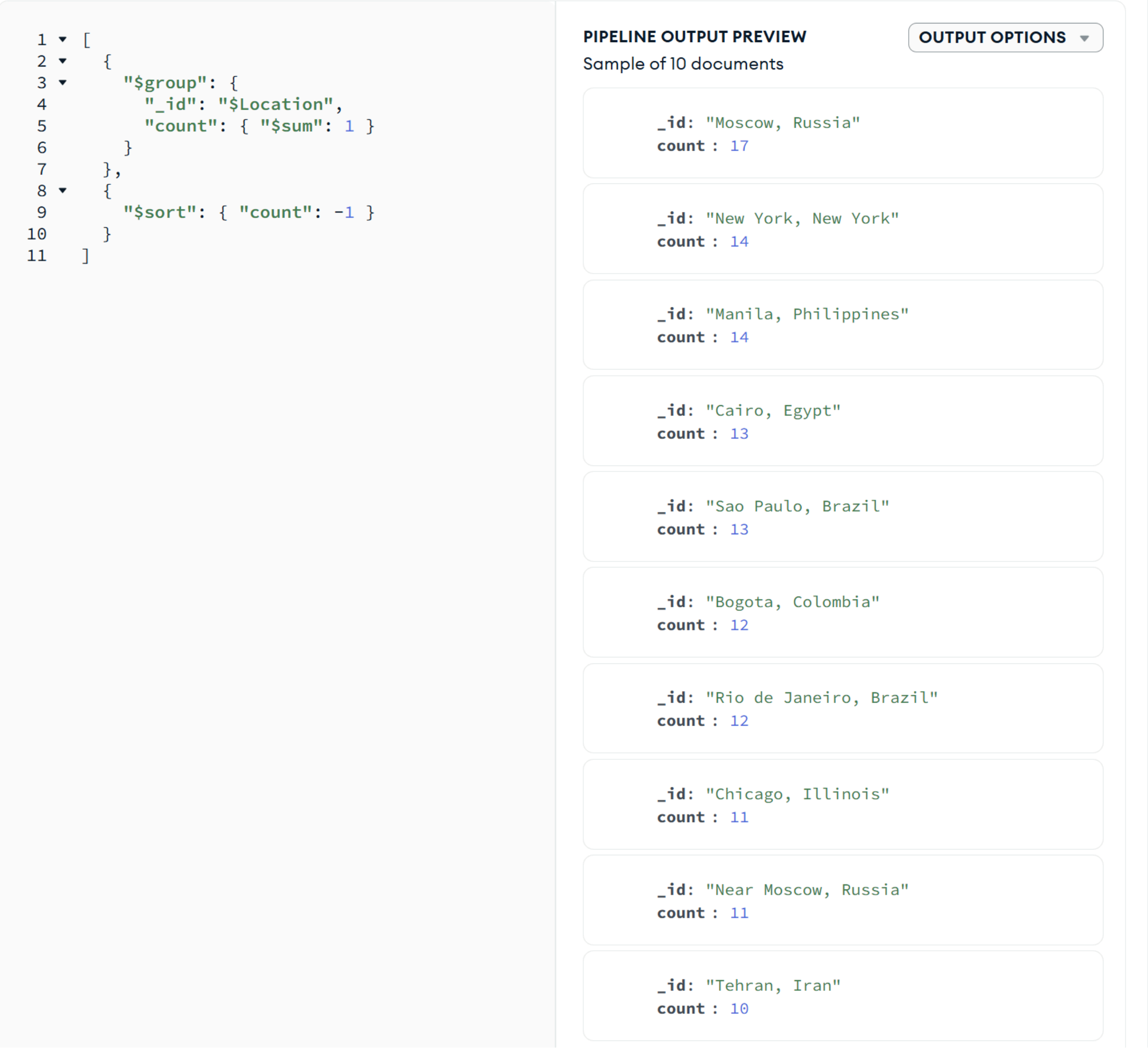
```
_id: "09/11/2001"
totalFatalities : 5890
```

Temporal Pattern questions reflection: It was fairly easy to utilize the LLM when answering these questions. I ran into issues with the first question, calculating the number of Fatalities by decade and looking for trends. I had to explicitly tell MongDB how to calculate the decades for the years, how to group data, and what the output should look like. Without doing this, MongoDB seemed to get a bit confused with how to extract the decade and do the calculations properly.
For question 2, the only issue was the (06:00 to 18:00) portion of the original question. MongoDB was a bit confused by this syntax, so I changed it to say “6am to 6pm” in order for it to understand how to create a proper query.
For question 3, I was able to use the question directly from the assignment in order to generate a query that solved the problem.

<u>Geographical Incident Analysis</u>

- **Number 1**
    - **Question:** High-Risk Locations
    - **Fields Used:** Location
    - **Goal:** Which specific cities, airports, or geographical areas have experienced the greatest number of incidents?

Prompt: Which specific cities, airports, or geographical areas have the most incidents?

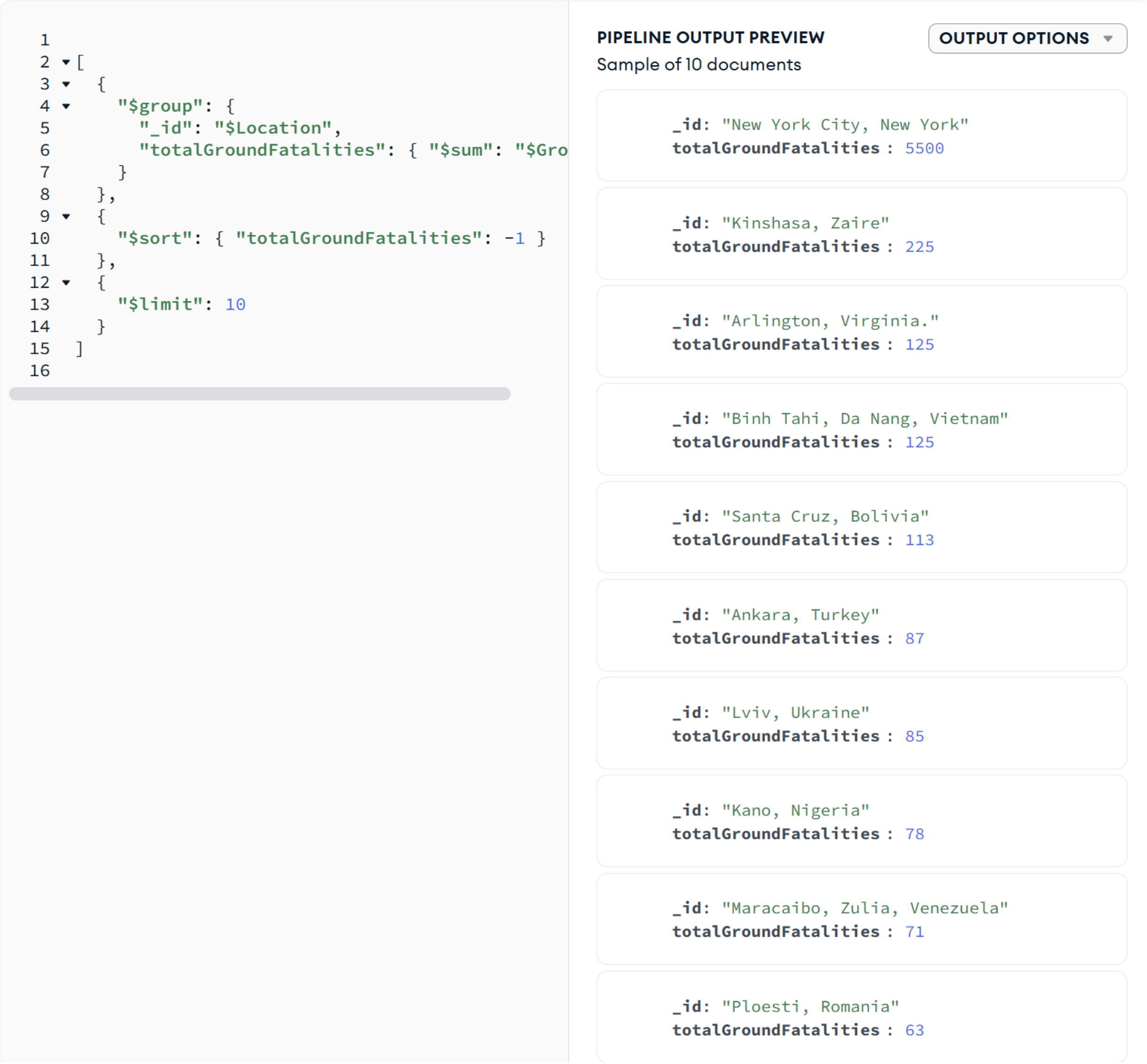


```
[
  {
    "$group": {
      "_id": "$Location",
      "count": { "$sum": 1 }
    }
  },
  {
    "$sort": { "count": -1 }
  }
]
```

```
_id: "Moscow, Russia"
count : 17

_id: "New York, New York"
count : 14

_id: "Manila, Philippines"
count : 14

_id: "Cairo, Egypt"
count : 13

_id: "Sao Paulo, Brazil"
count : 13

_id: "Bogota, Colombia"
count : 12

_id: "Rio de Janeiro, Brazil"
count : 12

_id: "Chicago, Illinois"
count : 11

_id: "Near Moscow, Russia"
count : 11

_id: "Tehran, Iran"
count : 10
```

- **Number 2**
    - **Question:** Ground Fatalities
    - **Fields Used:** Ground, Location
    - **Goal:** In what types of locations (e.g dense population vs remote area) do crashes result in the highest number of ground fatalities?

Prompt: In what types of locations (e.g dense population vs remote area) do crashes result in the highest number of ground fatalities?

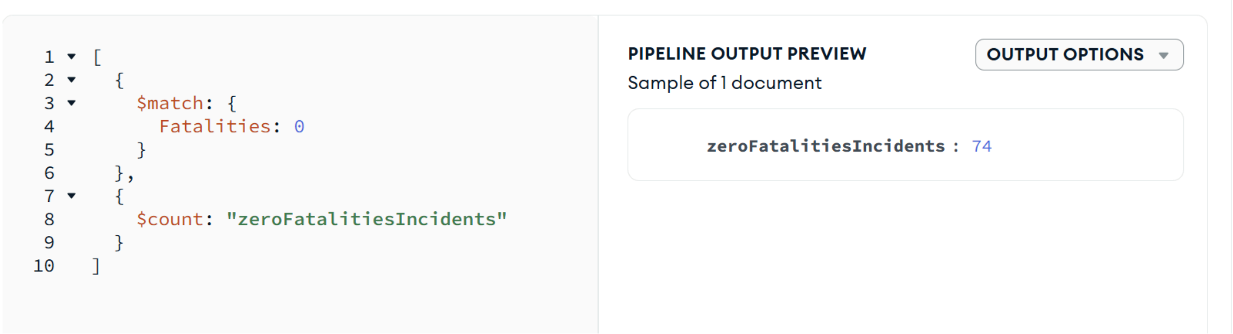


```
[
  {
    "$group": {
      "_id": "$Location",
      "totalGroundFatalities": { "$sum": "$Gro
    }
  },
  {
    "$sort": { "totalGroundFatalities": -1 }
  },
  {
    "$limit": 10
  }
]
```

```
_id: "New York City, New York"
totalGroundFatalities : 5500

_id: "Kinshasa, Zaire"
totalGroundFatalities : 225

_id: "Arlington, Virginia."
totalGroundFatalities : 125

_id: "Binh Tahi, Da Nang, Vietnam"
totalGroundFatalities : 125

_id: "Santa Cruz, Bolivia"
totalGroundFatalities : 113

_id: "Ankara, Turkey"
totalGroundFatalities : 87

_id: "Lviv, Ukraine"
totalGroundFatalities : 85

_id: "Kano, Nigeria"
totalGroundFatalities : 78

_id: "Maracaibo, Zulia, Venezuela"
totalGroundFatalities : 71

_id: "Ploesti, Romania"
totalGroundFatalities : 63
```

- **Number 3**
    - **Question:** Non-Fatal Incidents
    - **Fields Used:** Fatalities
    - **Goal:** How many incidents are recorded where the number of onboard fatalities was zero? (A measure of crash survivability).

Prompt: How many incidents are recorded where the number of onboard fatalities was zero?



```
[
  {
    $match: {
      Fatalities: 0
    }
  },
  {
    $count: "zeroFatalitiesIncidents"
  }
]
```

```
zeroFatalitiesIncidents : 74
```

For these questions, it was also quite easy to use the LLM to get meaningful answers.
For question 1, my original prompt included the word “greatest” instead of “most”. Using the word “greatest” gave me a query with a limit of one at the end since the LLM was assuming I was only looking for the largest number. Since the goal mentions “cities, airports, and areas” (all plural), I did not want this limit in place, which is why I adjusted the wording of my prompt to get more results.
For question 2, it seemed to be nearly impossible to have the query directly answer the question of dense populations versus remote areas. However, with the output that MongDB provided, we are clearly able to use our own reasoning to see what the top spots have in common. Clearly, dense population areas tend to have the most ground fatalities due to the large number of people in a given area.
For question 3, using the goal as a prompt worked perfectly and gave an output that answers the question fully.

Part 2: The LLM Challenge:

In this part, I will be intentionally trying to trip up the AI Query Generator built into MongoDB compass.

In order to do so, I re-imported the dataset and saved it as “airplane_crashes2”. This is because when I imported it the first time, all of my fields had proper data types. However, in order to cause a data type confusion, I needed certain fields (such as Fatalities and Abroad) to be strings instead. In airplane_crashes2, Fatalities and Abroad fields are both saved as strings.

Challenge Area	**LLM Weakness to Exploit**	**Initial Prompt**	**Expected Failure**
Data Type Confusion	Failure to convert string-based numbers (e.g. Fatalities) using $toInt before performing math.	Calculate each Operator’s average fatality rate and sort from highest percentage to lowest. Only output the Operator and the percentage.	Runtime Error (Data Type Error) I am expecting MongoDB to not change “Fatalities” and “Abroad” to be integers before attempting the calculations.
Ambiguous Context	Failure to correctly interpret a vague geographical or temporal request.	Where do most planes that crash take off from?	Logical error I think that MongoDB will generate a query that focuses on the Location field instead of looking at the first portion of the Route field (improper logic).
Complex Aggregation Structure	Failure to correctly sequence or nest multiple aggregation stages (e.g. $group then $match on the group results)	Treat the Fatalities field as a numeric value but be aware that some rows have it as "NULL" or null, which should be ignored. Use only the rows with real numeric Fatalities. I am interested in Operators that are associated with very severe crashes, where the fatalities reach at least 500 overall, and I want to know how many such Operators there are. Concentrate your calculation on the crashes that meet that level of severity and return a single number.	Runtime Error I expect MongoDB to have issues with knowing when to filter out the Null values and/or have issues with knowing when to do the grouping/filtering of the documents within the collection.

Data Type Confusion:

1. **Initial Prompt:** “Calculate each Operator’s average fatality rate and sort from highest percentage to lowest. Only output the Operator and the percentage.”
    - I received the following query, and after the query, I will analyze what happened:

```
[
  {
    $project: {
      Operator: 1,
      Fatalities: {
        $toInt: "$Fatalities"
      },
      AboardTotal: {
        $sum: [
          {
            $toInt: "$Aboard Crew"
          },
          {
            $toInt: "$Aboard Passangers"
          }
        ]
      }
    }
  },
  {
    $group: {
      _id: "$Operator",
      TotalFatalities: {
        $sum: "$Fatalities"
      },
      TotalAboard: {
        $sum: "$AboardTotal"
      }
    }
  },
  {
    $project: {
      Operator: "$_id",
      FatalityRate: {
        $cond: {
          if: {
            $gt: ["$TotalAboard", 0]
          },
          then: {
            $multiply: [
              {
                $divide: [
                  "$TotalFatalities",
                  "$TotalAboard"
                ]
              },
              100
            ]
          },
          else: 0
        }
      }
    }
  },
  {
    $sort: {
      FatalityRate: -1
    }
  },
  {
    $project: {
      _id: 0,
      Operator: 1,
      FatalityRate: 1
    }
  }
]
```

- MongoDB used the sum of “Aboard Crew” and “Aboard Passengers” instead of using the “Aboard” field, which is stored as a string.
- This output correctly attempted to change “Fatalities”, “Aboard Crew”, and “Aboard Passengers” to be integers using the $toInt command, but there was an error in doing so due to Null values:
    - “Executor error during aggregate command on namespace: examples.airplane_crashes2 :: caused by :: Failed to parse number 'NULL' in $convert with no onError value: Did not consume whole string.” As a result, there was no output of this query.
- In my next prompt, I tried telling MongoDB to directly divide Fatalities by Abroad in an attempt to have a true data type confusion error, hoping that MongoDB would not use the $toInt command.

2. **Modified prompt:** “Calculate each Operator’s average fatality rate by *directly dividing Fatalities by Abroad*, and sort from highest percentage to lowest. Only output the Operator and the percentage.”
    - I received the following query, and after the query, I will analyze what happened:

```
[
  {
    $addFields: {
      FatalitiesNumber: {
        $toInt: "$Fatalities"
      },
      AboardNumber: {
        $toInt: "$Aboard"
      }
    }
  },
  {
    $group: {
      _id: "$Operator",
      avgFatalityRate: {
        $avg: {
          $divide: [
            "$FatalitiesNumber",
            "$AboardNumber"
          ]
        }
      }
    }
  },
  {
    $project: {
      _id: 0,
      Operator: "$_id",
      avgFatalityRatePercentage: {
        $multiply: ["$avgFatalityRate", 100]
      }
    }
  },
  {
    $sort: {
      avgFatalityRatePercentage: -1
    }
  }
]
```

    - This output used Fatalities and Abroad as part of the calculation, but MongoDB still attempted to convert them to integers.
    - The same error occurred, however, due to null values:
        - “Executor error during aggregate command on namespace: examples.airplane_crashes2 :: caused by :: Failed to parse number 'NULL' in $convert with no onError value: Did not consume whole string.”
- In summary, I was unable to create a Data Type Confusion error, as MongoDB always knew to convert to integers before doing mathematical operations.

Ambiguous Context:

1. **Ambiguous Context Prompt**: Where do most planes that crash take off from?
    - The goal was to confuse MongoDB by asking for where the planes that crashed took off from, but also giving MongoDB the flexibility to pick the field that it thinks makes the most sense.
    - Location tells us where the plane crashed, but the Route field is needed to determine where the plane took off. In the below query, MongoDB mistakenly uses the Location field instead of the Route field:

```
[
 {
  $project: {
   Region: {
    $arrayElemAt: [
     {
      $split: ["$Location", ", "]
     },
     -1
    ]
   }
  }
 },
 {
  $group: {
   _id: "$Region",
   CrashCount: {
    $sum: 1
   }
  }
 },
 {
  $sort: {
   CrashCount: -1
  }
 },
 {
  $limit: 5
 }
]
```

2. **Working Prompt:** Find the top 10 departure locations for flights that crashed. Use the Route field as the flight path. Route is a string that often looks like "City A - City B" or "City A - City B - City C", but it sometimes has tab characters and inconsistent spacing, and some values are junk like "Training flight" with no dash at all. Treat only the records whose Route actually contains a dash as valid flight paths and ignore the rest. First

clean the text by removing tab characters and collapsing multiple spaces into a single space, and trim any leading or trailing whitespace. Then normalize the separator so every path uses a single " - " between segments. After that, treat the substring before the first " - " as the origin location of the flight. Count how many crashes originated from each origin, sort from the highest count to the lowest, and return only the top 10. Each result document should contain just the origin name and the number of crashes from that origin.

- The following query correctly uses the Route field, with direction from me. Data cleaning is performed as well, and I helped MongoDB understand how to know where each flight took off from. The output of the query helps us see that most planes that crash take off from Paris, London, Moscow, and New York City.

```
[
  {
    $match: {
      Route: {
        $regex: "-"
      }
    }
  },
  {
    $addFields: {
      cleanRoute: {
        $trim: {
          input: {
            $reduce: {
              input: {
                $split: [
                  {
                    $replaceAll: {
                      input: "$Route",
                      find: "\t",
                      replacement: " "
                    }
                  },
                  " "
                ]
              },
              initialValue: "",
              in: {
                $cond: [
                  {
                    $eq: ["$$this", ""]
                  },
                "$$value",
                {
                  $concat: [
                    "$$value",
                    {
                      $cond: [
                        {
                          $eq: ["$$value", ""]
                        },
                        "",
                        " "
                      ]
                    },
                    "$$this"
                  ]
                }
              ]
            }
          }
        }
      }
    }
  }
},
{
  $addFields: {
    origin: {
      $trim: {
        input: {
          $arrayElemAt: [
            {
              $split: ["$cleanRoute", " - "]
            },
            0
          ]
        }
      }
    }
  }
},
{
  $group: {
    _id: "$origin",
    count: {
        $sum: 1
      }
    }
  },
  {
    $sort: {
      count: -1
    }
  },
  {
    $limit: 10
  },
  {
    $project: {
      origin: "$_id",
      count: 1,
      _id: 0
    }
  }
]
```

Complex Aggregation Structure:

1. **Complex Aggregation Structure Prompt:** Treat the Fatalities field as a numeric value but be aware that some rows have it as "NULL" or null, which should be ignored. Use only the rows with real numeric Fatalities. I am interested in Operators that are associated with very severe crashes, where the fatalities reach at least 500 overall, and I want to know how many such Operators there are. Concentrate your calculation on the crashes that meet that level of severity and return a single number.
    - The goal of my request was to count “high-fatality Operators” whose total Fatalities reach at least 500 across all of their crashes. To solve this correctly, MongoDB would need to sum Fatalities per Operator and then filter based on that total.
    - The Fatalities field is stored as text, and some records contain “NULL” instead of a numeric value. My prompt warned MongoDB to avoid using those values and only treat real numeric Fatalities as valid data.
    - The query that MongoDB generated applied $toInt to every Fatalities values before checking whether values were null or “NULL”. As a result it attempted to concert these null values into a number, and when it did so, an error was triggered:
        - “Executor error during aggregate command on namespace: examples.airplane_crashes2 :: caused by :: Failed to parse number 'NULL' in $convert with no onError value: Did not consume whole string.”
    - In addition to causing this conversion failure, the query incorrectly filtered rows using Fatalities using “Fatalities > 499” before grouping. This only identifies operators with a single crash over 499 fatalities, rather than operators whose fatalities total at least 500 across multiple crashes.
    - Because of these incorrect aspects, the query does not produce an output, and even if it did run successfully, the query would still be logically incorrect.
    - The query that MongoDB generated is listed below:

```
[
  {
    $match: {
      $expr: {
        $and: [
          {
            $gt: [
              {
                $toInt: "$Fatalities"
              },
              499
            ]
          },
          {
            $ne: ["$Fatalities", null]
          },
          {
            $ne: ["$Fatalities", "NULL"]
          }
        ]
      }
    }
  },
  {
    $group: {
      _id: "$Operator",
      count: {
        $sum: 1
      }
    }
  },
  {
    $count: "severeCrashOperators"
  }
]
```

2. **Working Prompt:** Treat the Fatalities field as a numeric value, but some rows contain the literal text “NULL” or are actually null, so those should be ignored when working with Fatalities. Use only rows where Fatalities is a real number in text form. I’m interested in Operators linked to very severe crashes, where the total fatalities reach at least 500 overall. Focus only on those severe cases when determining the Operators, and return just one number that shows how many of these Operators there are.
    - This prompt works well because it subtly gives more clarity about the process of what must happen in the query in order to get a proper answer.
    - The following query allows us to see that there are 31 instances of Operators that go beyond this threshold.

```
[
  {
    $match: {
      Fatalities: {
        $regex: "^[0-9]+$"
      }
    }
  },
  {
    $addFields: {
      FatalitiesNumber: {
        $toInt: "$Fatalities"
      }
    }
  },
  {
    $group: {
      _id: "$Operator",
      TotalFatalities: {
        $sum: "$FatalitiesNumber"
      }
    }
  },
  {
    $match: {
      TotalFatalities: {
        $gte: 500
      }
    }
  },
  {
    $count: "SevereOperatorsCount"
  }
]
```