It is not unusual for students to analyse the dataset differently, model the databases in their own way, and formulate queries and present results using varied approaches. The examples provided from previous years illustrate the diversity of valid methods used for these tasks. These examples are for guidance only and should not be treated as definitive or prescriptive solutions. NOTE: These are only selected screenshots from longer reports and show only a partial view. A significant amount of explanation and descriptive content has been omitted.

An example of dataset analysis and reporting using bullet points

1 Activity 1: Data Analysis and Database Design

The provided data contains 106987 entries, which contain the following columns;

- *PatientId:* This uniquely identifies the patient. In the given dataset it is non-unique, showing that the same patient may have had multiple appointments over the period, with this count having a highest value of 88. There are 58994 distinct patients.
- *AppointmentId:* This uniquely identifies the appointment, and is unique over the dataset. This can therefore be used as the primary key.
- *Gender:* The gender of the patient, either male or female. The dataset has 39081 distinct female patients, and 21189 male patients. How this affects the overall no-show rate is explored in Query J of Activity 2.
- *ScheduledDay:* The day that the appointment was scheduled on, from 2015-11-10 to 2016-06-08, a period of just under 7 months.
- *AppointmentDay:* The day that the appointment was scheduled for, ranging from 2016-04-29 to 2016-06-08, a period of just over a month.
- *Age:* The age of the patient, in a range of 1 to 115. This can vary between appointments for the same patient, suggesting that the patient had birthdays between appointments. A frequency graph is shown in figure 1 showing that after an age of 1, the frequency of ages remains roughly consistently between 1000 and 1750 until around an age of 65 where the frequency starts dropping quickly. How this relates to the overall no-show rate is explored in Query B in Activity 2, which gives a maximum difference in no-show rate of 9.83%. This shows that this is a significant aspect of this dataset.
- *Neighbourhood:* The neighbourhood of the patient. This does not vary between appointments for the same patient, and there are 81 distinct neighbourhoods. A count of patients per neighbourhood is done in activity A of Activity 2 and a count of total no-shows is done in Query E of Activity 2. These indicate that the neighbourhood does have a highly significant influence of attendance rate - even after focussing on neighbourhoods with more than 100 patients the no-show rate can vary by up to 14.7%.
- *Scholarship:* A boolean representing if the patient had a scholarship, such as for healthcare coverage. This does not vary between appointments for the same patient. There are 54513 patients without a scholarship and 5757 with a scholarship, meaning that 5.4% of patients

An example of dataset analysis and reporting using a numbered list

2.1 Dataset Overview

The dataset includes medical appointment information, capturing various factors potentially influencing patient attendance. Key columns include:

1. **PatientId**: Unique identifier for each patient, which can be used to track individual attendance history.
2. **AppointmentId**: Unique identifier for each appointment, ensuring each instance is recorded separately.
3. **Gender**: Patient gender, represented as 'F' or 'M'. Gender-based attendance patterns can be assessed.
4. **ScheduledDay**: Date the appointment was scheduled, useful for calculating lead time before the appointment.
5. **AppointmentDay**: Date of the appointment. The difference between ScheduledDay and AppointmentDay (captured in Date.diff) provides insights into whether the timing affects attendance.
6. **Age**: Patient age, relevant for analysing no-show rates across age groups (e.g., under 18, 18-30, 31-50, Over 50).
7. **Neighbourhood**: Patient's Neighbourhood, allowing analysis of no-show rates by geography.
8. **Scholarship**: Boolean field indicating if the patient has a financial aid scholarship, useful for comparing attendance rates between scholarship and non-scholarship patients.
9. **Hipertension, Diabetes, Alcoholism**: Boolean indicators of chronic conditions, enabling analysis of attendance patterns among patients with specific health conditions.

An example of dataset analysis and reporting using a tabular format

Overview of the Dataset

The dataset contains information about patient appointments in a healthcare setting. It includes details about patients, their medical conditions, and the outcomes of their appointments. By splitting the dataset into four distinct tables—patients, appointments, conditions, and neighbourhoods—we have normalized the data, ensuring better structure, maintainability, and scalability.

Table 1: Patients

Column Name	Data Type	Description	Significance	Range/Format
PatientId	VARCHAR/String	A unique identifier for each patient. This serves as the primary key and links this table to others.	Crucial for tracking patients across multiple appointments and conditions.	Numeric, represented in exponential notation in the source data (e.g., 2.98725E+13), but stored as a string for accuracy.
Gender	CHAR/String	Represents the gender of the patient, either 'M' (Male) or 'F' (Female).	Useful for demographic analysis and identifying patterns by gender.	'M' or 'F'
Age	INTEGER	The age of the patient.	Allows segmentation of appointments and health conditions by age groups.	0 (newborns) to a plausible maximum, such as 120.
NeighbourhoodId	INTEGER (Foreign Key)	A reference to the neighbourhoods table for the patient's area of residence.	Facilitates geographical analysis.	N/A

Table 2: Neighbourhoods

Column Name	Data Type	Description	Significance	Range/Format
Neighbourhood	VARCHAR/String	Name of the geographical location where the patient resides.	Allows grouping of data by geographical location for spatial analysis.	Text
NeighbourhoodId	INTEGER (Primary Key)	Unique identifier for each neighbourhood.	Provides a unique reference for each location.	N/A

An example of dataset analysis and reporting supported by screenshots

Columns

Patient ID

Each value within this column, represents a Unique identifier for a patient. Within this dataset, there are currently duplicate values as patients are able to book multiple appointments, however, each patient will only have one Patient ID.

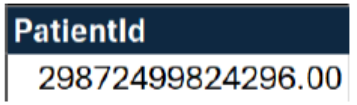


Figure 1 Shows The Patient ID Column Identifier, With The first Value Beneath

All values within this column are whole numbers, the largest value (shown in figure 2) , disregarding the decimal and the digits that come after, is a total of 15 digits long.

Figure 2 Shows The Largest Number Present in the Patient ID column, Ranging from A2, to A106988. Formula used to obtain this: =MAX(A2:A106988)

The most appropriate data type for this, in SQL, would be the BIG INT. The reasoning for this is as follows:

- Capable of storing 64-bit numbers.
- Int data type cannot store the numbers present in the Patient ID column.
- The largest number the BIG INT data type can store can have a total of 19 digits with precision.
- All Patient ID's are whole numbers.

In MongoDB, there is no BIGINT datatype, however, there is an equivalent that has the same function. This data type is called "Long", possessing the same functionality as BIGINT.

Appointment ID

Similar to Patient ID, values within this column represent a unique identifier for an Appointment between a Medical Specialist and a Patient. Although, their similarities stop here, as there are no duplicates within this column, each appointment has a different number assigned to it.

Figure 3 Shows The Largest Number Present in the Appointment ID column, Ranging from B2, to B106988. Formula used to obtain this: =MAX(B2:B106988)

The largest value within this column is shown in figure 3.

Disregarding decimal digits, it is a total of 7 digits long. In SQL, the most appropriate data type is an Integer, the reasoning is as such:

Another example of dataset analysis and reporting

1.1 Dataset Description and Analysis

The dataset contains 106,987 rows and 15 columns. The attributes consist of:

PatientId

Data Type: Integer

Range of Values: Sequence of positive integers.

A unique identification assigned to every patient. It is necessary for correlating multiple appointments to particular patients and allows for monitoring patient behaviour. Enabling examination of specific patterns regarding individual patients and keeping track of a patient's demographics, past appointments, and other relevant information. This attribute is essential for combining and aggregating patient-related data, even if it has no direct bearing on no-show trends. Frequent absences from particular patient IDs may be an indication of systemic problems, such as conflicts with scheduling or accessibility difficulties.

AppointmentID

Data Type: Integer

Range of Values: Sequence of positive integers.

A unique identification issued to every healthcare appointment. Makes certain that each patient's data is individually identified, which is critical for analysing individual appointment attendance statistics. This attribute facilitates the analysis of trends over recurring visits, including missed appointment patterns for individual patients. Patterns such as repeated cancellations or missed follow-ups can be found by connecting other data fields to this ID. Comparing appointments with comparable features, like scheduling or reminders, may also uncover trends.

Gender

Data Type: String

Range of Values: 'M' for male, 'F' for female.

Denotes the patient's gender. Enables for the investigation of how gender influences the likelihood of no-shows, as well as insights regarding gender-based health care utilisation. The

Some students used graphs and charts to present the insights they drew from the dataset.

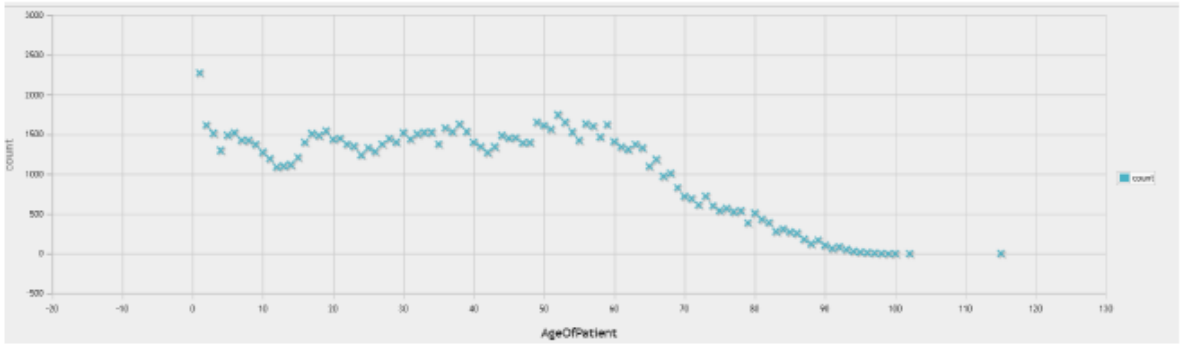
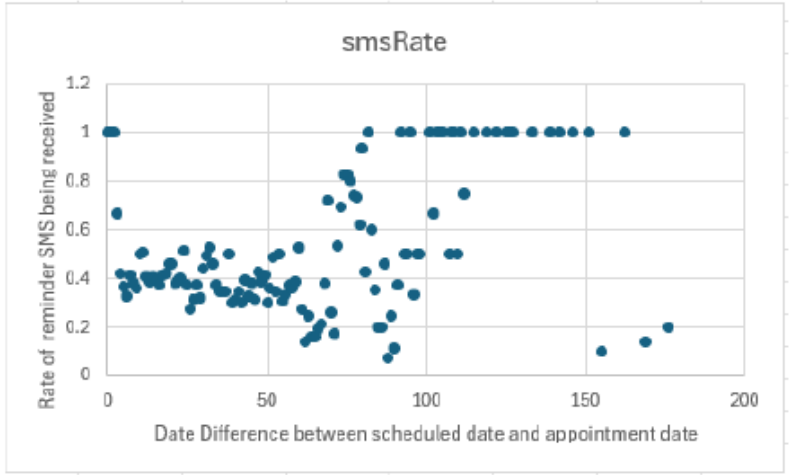


Figure 1: A frequency graph of the age of patients.



Some students presented their insights in a purely textual format.

Insights and Analysis

1. Patient Demographics and No-Show Patterns

- **Age:**
  Younger patients may have higher no-show rates due to fewer chronic conditions, while seniors may struggle with mobility or transportation. Tailored strategies like frequent reminders could help younger patients, while seniors may benefit from transportation assistance.
- **Gender:**
  Comparing no-show rates by gender could reveal if societal or economic factors disproportionately affect attendance for men or women.

2. Impact of Neighbourhood and Socioeconomic Background

- **Neighbourhood Trends:**
  Areas with higher no-show rates might indicate barriers like transportation or healthcare access. Targeted outreach, such as mobile health services, could address these issues.
- **Scholarship Indicator:**
  Patients receiving financial assistance may face additional challenges, like work constraints or childcare. Programs like subsidized transport or extended hours could reduce no-shows.

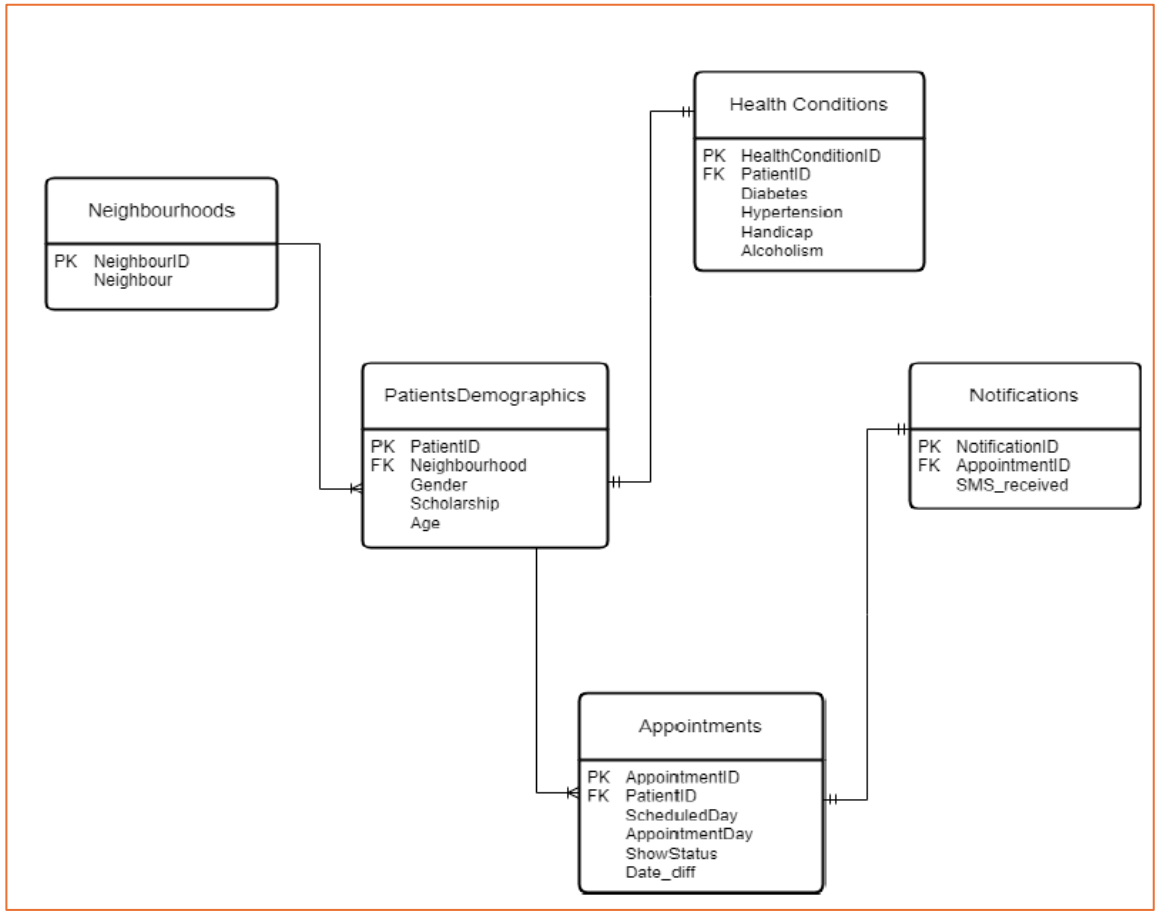
3. Medical Conditions and Appointment Attendance

- **Chronic Conditions:**
  Patients with conditions like hypertension and diabetes require regular visits, and higher no-show rates could indicate misunderstanding or access issues. Interventions like condition-specific reminders could improve attendance.
- **Alcoholism and Disabilities:**
  These patients might face stigma or mental health barriers, so more inclusive environments and tailored support can help.

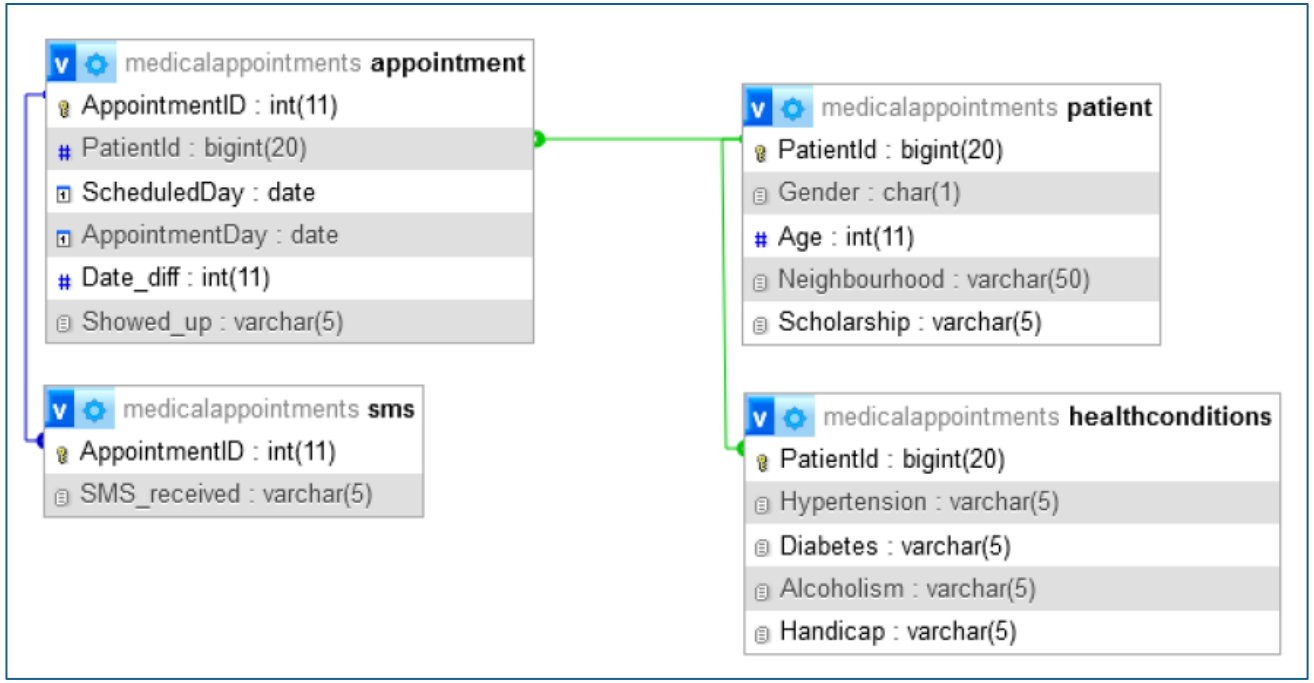
4. Effectiveness of SMS Reminders

- **Role of Reminders:**
  Analyzing the impact of SMS reminders on attendance can guide clinics in expanding reminder programs.
- **Timing of Reminders:**
  Determining the optimal time for reminders, such as 1–2 days before appointments, could improve show-up rates.

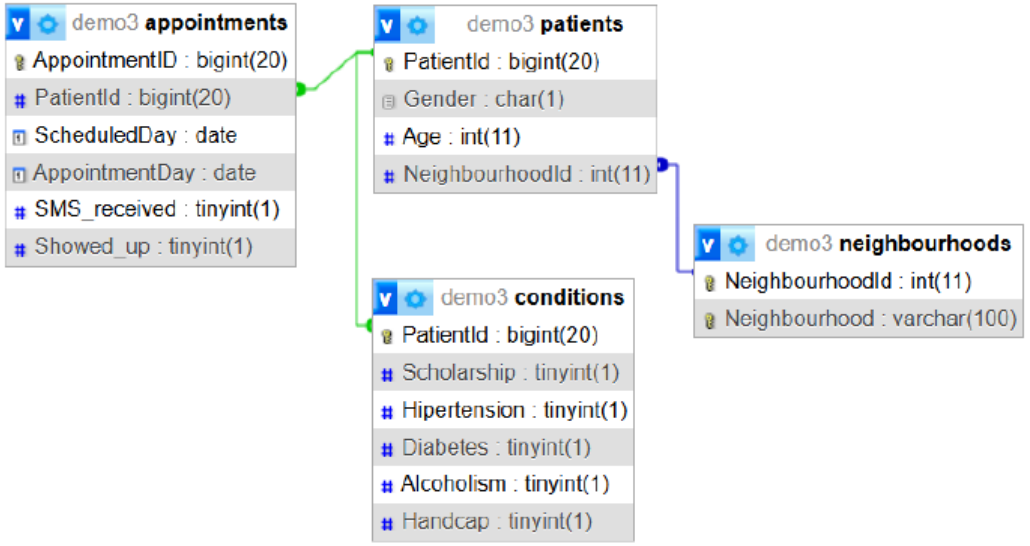
In the previous year, students explained their relational database modeling using detailed textual descriptions of the normalization process (e.g., 1NF → 2NF → 3NF), and then presented the final ERD.



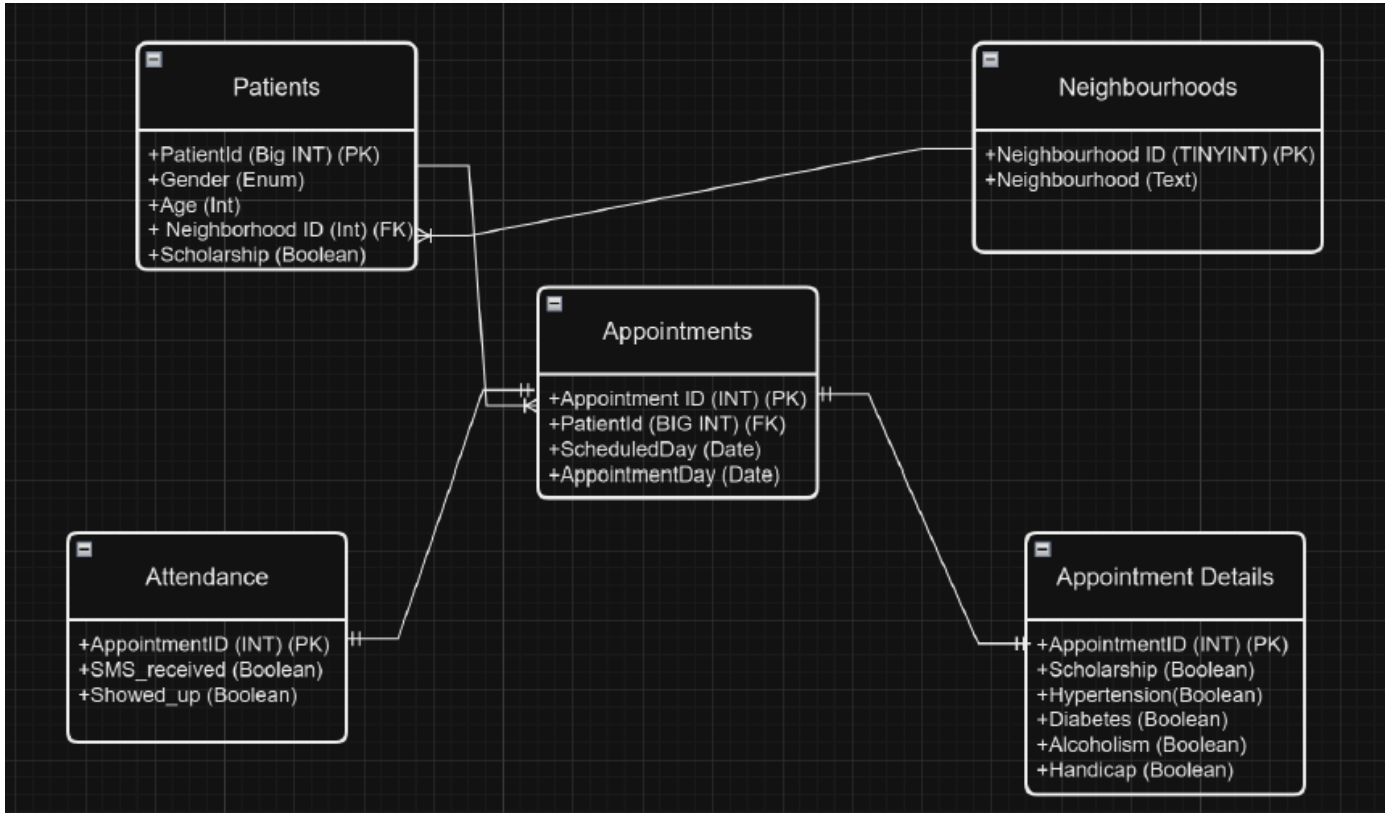
Example of an ERD included by one student



Example of an ERD included by one student



An example of an ERD included by one student



An example of an ERD included by one student

```
{
    "AppointmentId": 5642903,
    "Patient": {
        "PatientId": "2.98725E+13",
        "Gender": "F",
        "Age": 62,
        "Neighbourhood": "JARDIN DE PENHA",
        "Scholarship": false,
        "HealthConditions": {
            "Hypertension": true,
            "Diabetes": false,
            "Alcoholism": false,
            "Handcap": false
        }
    },
    "ScheduledDay": "29/04/2016",
    "AppointmentDay": "29/04/2016",
    "Date.diff": 0,
    "SMS_received": false
    "Showed_up": true
}
```


An example of a MongoDB document model submitted by one student


```
{
    _id : Int64,
    Patient{
        PatientId : Int64,
        Age: Int32,
        Gender :  String,
        Neighbourhood: String,
        AdditionalInformation: Array
    }
    DateScheduledOn : DateTime,
    AppointmentDate: DateTime,
    SMSReceived: Bool,
    ShowedUp: Bool
}
```


Another example of a MongoDB document model submitted by another student

For the SQL we use the following query:

```
SELECT Neighbourhood, COUNT(Neighbourhood) as Count
FROM patients p
GROUP By Neighbourhood
ORDER BY Count ASC
```

For the MongoDB data, we use the following aggregation query:

```
[
  {
    $project: {
      _id: 0,
      "Patient.PatientId": 1,
      "Patient.Neighbourhood": 1
    }
  },
  {
    $group: {
      _id: "$Patient.PatientId",
      Neighbourhood: {
        $first: "$Patient.Neighbourhood"
      }
    }
  },
  {
    $group: {
      _id: "$Neighbourhood",
      count: {
        $count: {}
      }
    }
  },
  {
    $sort: {
      count: 1
    }
  }
]
```

For the MySQL and MongoDB queries and the comparison of results, different students adopted different styles.

- This student presented each SQL query followed immediately by its equivalent MongoDB query, and then displayed the results side by side (see next page).

Neighbourhood	Count ▲ 1
PARQUE INDUSTRIAL	1
ILHAS OCEÂNICAS DE TRINDADE	2
ILHA DO FRADE	5
AEROPORTO	7
ILHA DO BOI	22
PONTAL DE CAMBURI	45
MORADA DE CAMBURI	66
NAZARETH	83
SEGURANÇA DO LAR	87
UNIVERSITÁRIO	92
HORTO	94
SANTA HELENA	102
FRADINHOS	126
ENSEADA DO SUÁ	138
COMDUSA	146
ANTÔNIO HONÓRIO	169
DE LOURDES	172
ARIOVALDO FAVALESSA	173
BOA VISTA	181
MÁRIO CYPRESTE	198
BARRO VERMELHO	213
DO MOSCOSO	218
SANTA LÚCIA	222
SANTA LUÍZA	225
SANTA CECÍLIA	240

```
_id: "PARQUE INDUSTRIAL"
count : 1

_id: "ILHAS OCEÂNICAS DE TRINDADE"
count : 2

_id: "ILHA DO FRADE"
count : 5

_id: "AEROPORTO"
count : 7

_id: "ILHA DO BOI"
count : 22

_id: "PONTAL DE CAMBURI"
count : 45

_id: "MORADA DE CAMBURI"
count : 66

_id: "NAZARETH"
count : 83

_id: "SEGURANÇA DO LAR"
count : 87

_id: "UNIVERSITÁRIO"
count : 92
```

The results of the SQL and MongoDB queries were presented side by side, making comparison easier.

Following these figures, students provided descriptive explanations of the results along with their own reflections in plain text.

```sql
SELECT
    CASE
        WHEN p.Age < 18 THEN 'Under 18'
        WHEN p.Age BETWEEN 18 AND 30 THEN '18-30'
        WHEN p.Age BETWEEN 31 AND 50 THEN '31-50'
        ELSE 'Over 50'
    END AS AgeGroup,
    COUNT(a.AppointmentID) AS TotalAppointments,  -- Total appointments for each age group
    SUM(CASE
            WHEN (a.Showed_up = 'FALSE' OR a.Showed_up IS NULL) THEN 1
            ELSE 0
        END) AS NoShows,  -- Handle 'FALSE' or NULL as no-show
    ROUND(
        (SUM(CASE
                WHEN (a.Showed_up = 'FALSE' OR a.Showed_up IS NULL) THEN 1
                ELSE 0
            END) / COUNT(a.AppointmentID)) * 100,
        2
    ) AS NoShowPercentage
FROM Appointment a
JOIN Patient p ON a.PatientId = p.PatientId
GROUP BY AgeGroup;
```

AgeGroup	TotalAppointments	NoShows	NoShowPercentage
18-30	18245	4491	24.61
31-50	29467	6100	20.70
Over 50	35417	5726	16.17
Under 18	23858	5363	22.48

- In this case, the SQL query was first shown as a screenshot, followed immediately by its result. Then, the MongoDB version of the same query and its corresponding result were presented (see next page).

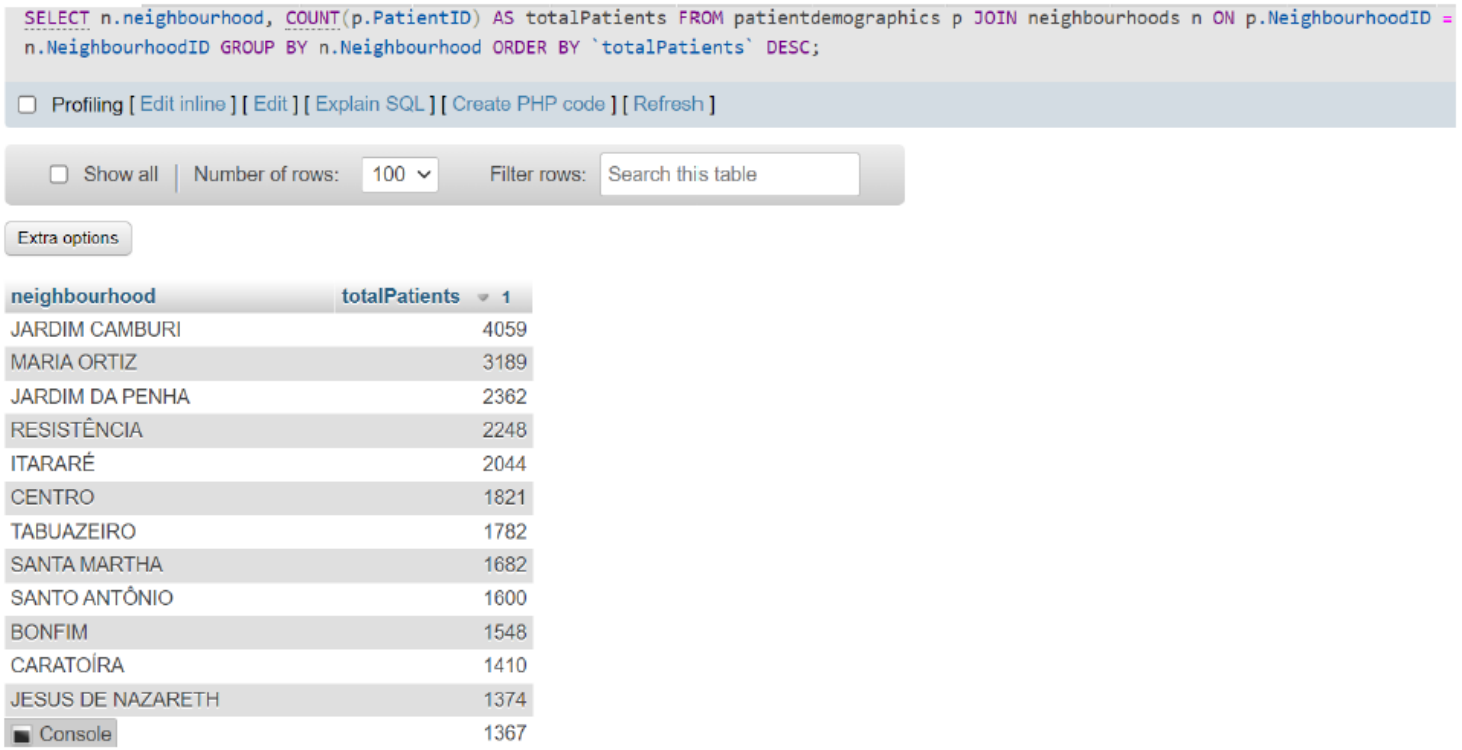
```
db.medicalappointments.aggregate([
  {
    $addFields: {
      AgeGroup: {
        $switch: {
          branches: [
            { case: { $lt: ["$Age", 18] }, then: "Under 18" },
            { case: { $and: [{ $gte: ["$Age", 18] }, { $lte: ["$Age", 30] }] }, then: "18-30" },
            { case: { $and: [{ $gte: ["$Age", 31] }, { $lte: ["$Age", 50] }] }, then: "31-50" },
          ],
          default: "Over 50"
        }
      }
    }
  },
  {
  percentage
    $group: {
      _id: "$AgeGroup",  // Group by AgeGroup
      TotalAppointments: { $sum: 1 },
      NoShows: {
        $sum: {
          $cond: [
            { $or: [{ $eq: ["$Showed_up", false] }, { $eq: ["$Showed_up", null] }] },
            1,
            0
          ]
        }
      }
    }
  },
  {
    $addFields: {
      NoShowPercentage: {
        $round: [
          { $multiply: [{ $divide: ["$NoShows", "$TotalAppointments"] }, 100] }, 2
        ]
      }
    }
  },
  {
    $sort: { _id: 1 }
  }
])
```

```
{
  _id: '18-30',
  TotalAppointments: 18252,
  NoShows: 4493,
  NoShowPercentage: 24.62
}
{
  _id: '31-50',
  TotalAppointments: 29472,
  NoShows: 6102,
  NoShowPercentage: 20.7
}
{
  _id: 'Over 50',
  TotalAppointments: 35423,
  NoShows: 5727,
  NoShowPercentage: 16.17
}
{
  _id: 'Under 18',
  TotalAppointments: 23840,
  NoShows: 5358,
  NoShowPercentage: 22.47
}
```

The MongoDB query and the results it returned.

i. How many patients are registered in each neighbourhood?

```
SELECT
    n.neighbourhood,
    COUNT(p.PatientID) AS totalPatients
FROM
    patientdemographics p
JOIN
    neighbourhoods n ON p.NeighbourhoodID = n.NeighbourhoodID
GROUP BY
    n.Neighbourhood
ORDER BY
    `totalPatients` DESC;
```



- This student presented the SQL query (both as text and as a screenshot) along with its result immediately afterward.
- The same approach was then used for the MongoDB version of the query (see next page).

i. How many patients are registered in each neighbourhood?

```
db.healthcare.aggregate([{$group: {_id: "$neighbourhood", UniquePatients: {$addToSet: "$_id"}}},
{$project: {_id: 0, Neighbourhood: "$_id", RegisteredPatients: {$size: "$UniquePatients"}}}, {$sort:
{RegisteredPatients: -1}}])
```

```
db.appointments.aggregate([{$group: {_id: "$neighbourhood", UniquePatients: {$addToSet: "$_id"}}}, {$project: {_id: 0, N
{
  Neighbourhood: 'JARDIM CAMBURI',
  RegisteredPatients: 4059
}
{
  Neighbourhood: 'MARIA ORTIZ',
  RegisteredPatients: 3189
}
{
  Neighbourhood: 'JARDIM DA PENHA',
  RegisteredPatients: 2362
}
{
  Neighbourhood: 'RESISTÊNCIA',
  RegisteredPatients: 2248
}
{
  Neighbourhood: 'ITARARÉ',
  RegisteredPatients: 2044
```

The MongoDB version of the query and the results it produced.