Master in Artificial Intelligence
Academic Year 2025-2026

November 2025

Data Engineering (P4251102)

Dimensional modelling and ETL data processing: Medical Imaging

Yago Estévez Figueiras[1],
Andrea Real Blanco[2],
Francisco Manuel Vázquez Fernández[3]

Abstract. The objective of this lab practice is to implement a Dimensional Model to support the analysis of medical images. Specifically, we target the analysis of radiology images pertaining to patients diagnosed with Lung Adenocarcinoma, the most prevalent form of lung cancer. The Star Schema is designed to provide insights into the following dimensions: imaging quality, equipment performance, and protocol consistency as measured across time, patients, and imaging stations. A complete ETL pipeline was developed to extract, transform, and load the raw data into the MongoDB-based data warehouse.

Contents

[1]yago.estevez.figueiras@rai.usc.es
[2]andrea.real@rai.usc.es
[3]franciscomanuel.vazquez.fernandez@rai.usc.gal

1 Software and Requirements

To reproduce the pipeline and populate the data warehouse, the following software and steps are required:

1. Non-relational Database: MongoDB and MongoDB Compass are used as the DBMS.
2. Install the required Python packages (Recommended Python version 3.x):

   ```
   pip install -r requirements.txt
   ```

3. Run `pipeline.ipynb` to execute the ETL process and populate the MongoDB database. The database connection parameters and data directory paths are managed through the variables `MONGO_URI`, `DB_NAME`, `DICOM_PATH`, and `JPEG_PATH` in the Python code.

2 Data Analysis and Modelling

This section provides an overview of the source data characteristics, explains the analytical purpose of the project, and presents the provided star schema.

2.1 DICOM data

The primary source for this analysis is a collection of radiology images and their associated metadata, focusing on Lung Adenocarcinoma patients. The data is stored in the **DICOM** (Digital Imaging and Communications in Medicine) format, which is the global standard for medical image storage, transmission, and printing. A DICOM file contains both the image pixel data and a wealth of structured metadata.

The collection consists of radiology images of the type **Computed Tomography (CT) scans**. These scans use X-rays to create detailed cross-sectional images of the body, allowing for the visualization of lung tumors and the assessment of cancer progression.

The structured metadata accompanying the pixel data provides context for the image acquisition. It includes information about the patient, such as age and sex; details about the imaging station, including the manufacturer and model; and various image properties, such as slice thickness and pixel spacing.

Some DICOM attributes used in the analysis might not be immediately self-explanatory, so we provide a short overview to clarify their purpose: The **contrast agent** refers to a substance administered to the patient to enhance the visibility of specific tissues or blood vessels during the scan. The **patient position** describes the patient's orientation in the scanner using standardized DICOM codes: the first letter indicates head-first (H) or feet-first (F) entry, the second letter is usually F by convention, and the third letter indicates the detailed orientation such as supine (S, lying on the back), prone (P, lying on the stomach), or decubitus left/right (DL/DR, lying on the left or right side). The **slice thickness** represents the depth, measured in millimeters, of each cross-sectional image captured by the CT scanner; smaller slice thickness values provide higher spatial resolution and more detailed three-dimensional reconstructions, though they also increase the data volume. The **photometric interpretation** defines how pixel intensity values are translated into image brightness, indicating whether higher pixel values correspond to denser (brighter) or less dense (darker) tissues. The **tube current** (measured in milliamperes, mA) specifies the electrical current applied to the X-ray tube during acquisition, with higher values reducing image noise at the cost of increased radiation exposure. Finally, the **exposure time** (measured in milliseconds, ms) indicates the duration of the X-ray emission for each slice, and together with tube current determines the total radiation dose delivered during the scan.

2.2 Analytical Objectives and Justification

The analytical model is designed to support queries that explore relationships between imaging parameters, equipment characteristics, and patient context. By structuring the data using a dimensional model, it becomes possible to perform aggregated analyses and identify patterns across different dimensions of the dataset.

Specifically, the model should enable queries such as:

1. What is the average exposure time and tube current for each manufacturer and model?
2. Which body part requires the longest average exposure time?
3. What is the variation in average slice thickness across different scanner models for chest studies?
4. How does the average age differ between patients requiring contrast media and those who do not?
5. How has the volume of studies changed over time for each equipment manufacturer?

2.3 Dimensional Model Diagram

The dimensional model consists of a single star schema with the fact table at its center and five surrounding dimension tables (see Figure 1).

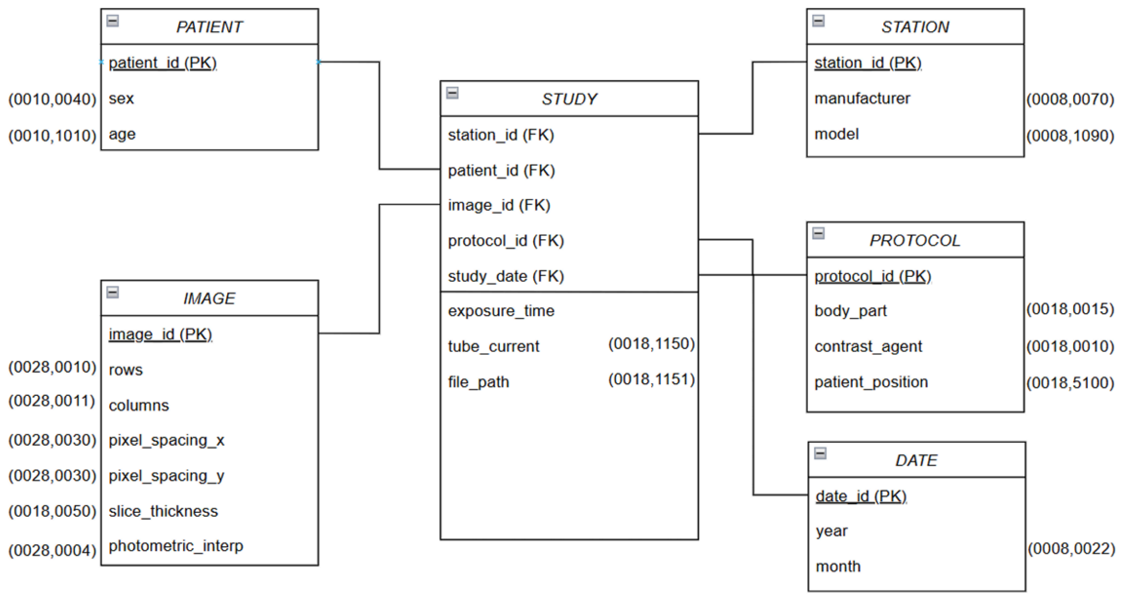


Figure 1: Star schema of the medical imaging dimensional model.

The **fact table** (`fact_study`) contains the measures `exposure_time` (DICOM tag: 0018,1150), `tube_current` (0018,1151), and the JPEG image path `file_path`, along with the foreign keys to the five dimensions.

The **dimension tables** contain attributes that provide additional context:

- **Patient** (`dim_patient`): Stores patient demographics `sex` (0010,0040) and `age` (0010,1010).
- **Station** (`dim_station`): Stores equipment metadata `manufacturer` (0008,0070) and `model` (0008,1090).

- **Protocol** (`dim_protocol`): Stores procedural parameters `body_part` (0018,0015), `contrast_agent` (0018,0010), `patient_position` (0018,5100).
- **Date** (`dim_date`): Stores temporal components `year` and `month` derived from the study date (0008,0022).
- **Image** (`dim_image`): Stores physical image properties `rows` (0028,0010), `columns` (0028,0011), `pixel_spacing` (0028,0030), `slice_thickness` (0018,0050), and `photometric_interp` (0028,0004).

Each dimension table uses a surrogate key in the form of a hash, generated from the corresponding attribute dictionaries, to uniquely identify each row independently of the original DICOM identifiers. The fact table references these surrogate keys as foreign keys, linking each study record to the corresponding patient, station, protocol, date, and image attributes.

2.4 Key Data Analysis

Initial data inspection was performed to validate the source data's quality and distribution. The dataset, consisting of 100 DICOM files, exhibits a high degree of consistency, suggesting it is a curated subset from the TCGA-LUAD dataset, which aligns with the Kaggle's dataset description.

The initial null value check (Listing 1) confirms that nearly all DICOM metadata tags required for the dimensional model are present for every file. Furthermore, the string-based fields maintain uniform formatting across all records. For instance, the `StudyDate` string is consistently represented as "YYYYMMDD", and `PatientAge` is consistently given in years (e.g., "061Y"). The only "missing" data point is the `ContrastAgent` field, which is absent in 86 out of 100 DICOM files. In the medical domain, this omission is customary and does not indicate unknown information; rather, it means that the study was performed without the use of a contrast agent.

Visualizing some dimensional attributes shows that the dataset profile aligns with what would be expected in a study focused on Lung Adenocarcinoma: The distribution of patient ages (Figure 2) is centered around an older demographic and is equally present in both sexes, which is consistent with the primary age range for this specific cancer diagnosis. We noted that while the dataset contains 100 images, these belong to only 54 unique patients. Importantly, the sex distribution across these 54 unique patients (53.70% male, 46.30% female) closely mirrors the distribution observed across all 100 images. This confirms that the overall cohort characteristics are representative. An analysis of study acquisition dates can be seen in Figure 3.

```
--- Validation: Null Values ---
ContrastAgent       86
PatientID            0
file_path            0
PatientAge           0
Manufacturer         0
Model                0
PatientSex           0
StudyDate            0
Rows                 0
PixelSpacing         0
Columns              0
SliceThickness       0
PhotometricInterp    0
BodyPart             0
PatientPosition      0
```

Listing 1: Distribution of null values across the dataset.

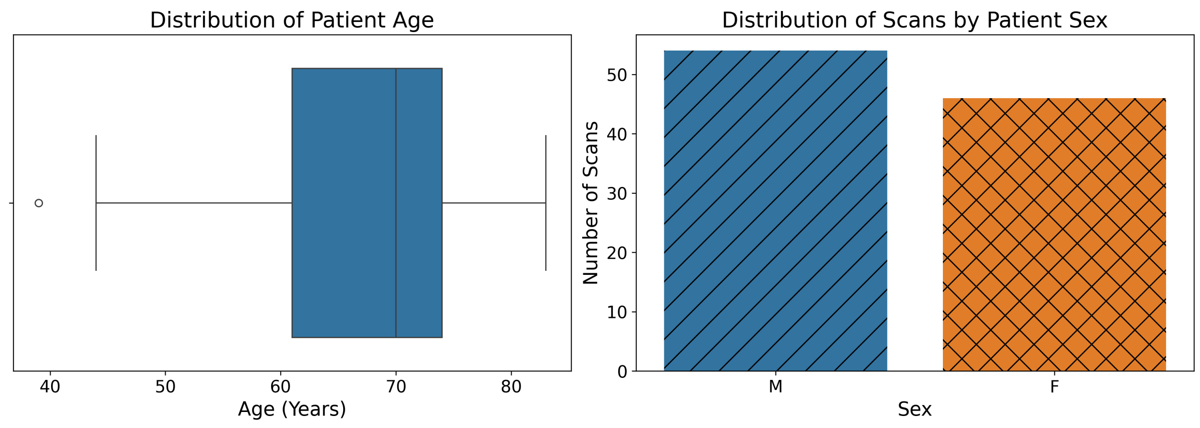


Figure 2: Distribution of patient ages and sex across the dataset.

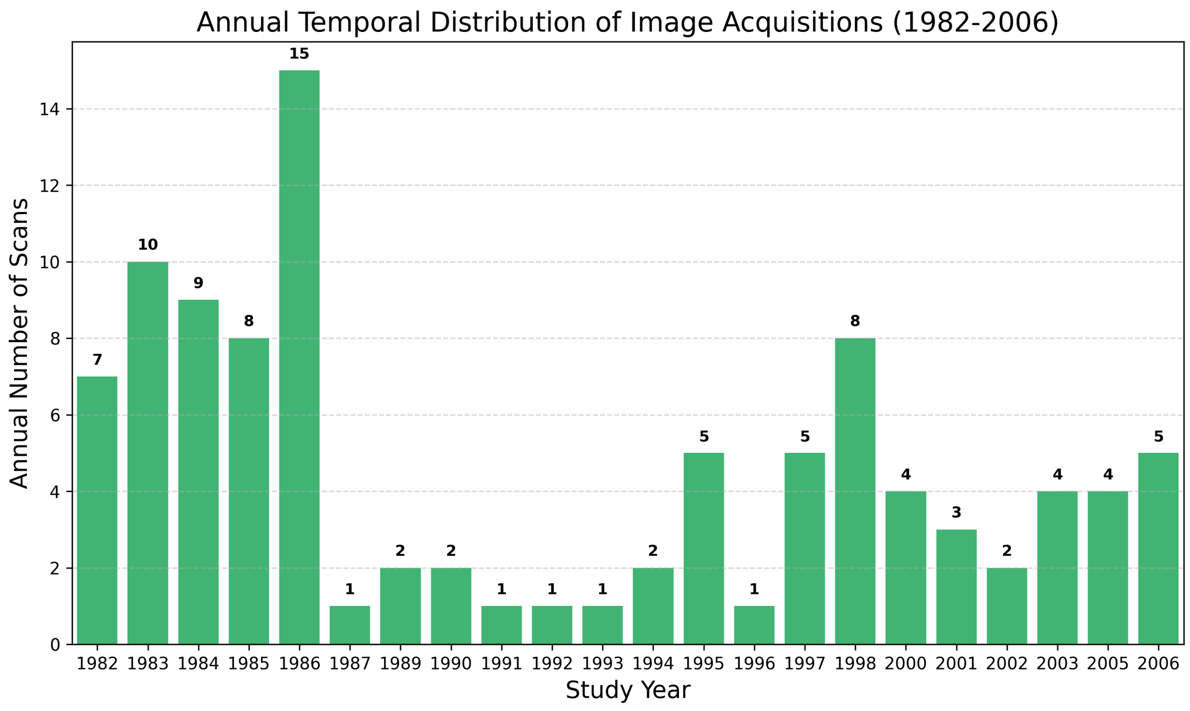


Figure 3: Distribution of DICOM files over time, showing the temporal coverage of the dataset.

3 Data Pipeline (ETL)

The ETL pipeline is implemented in Python and orchestrated within the `pipeline.ipynb` notebook. The pipeline processes the raw DICOM metadata and image data from Kaggle's dataset CT Medical Images, applies necessary transformations and cleansing rules, and loads the final dimensional model into the MongoDB data warehouse.

The **extraction phase** involves reading both structured metadata and unstructured pixel data from the DICOM files. The raw data is extracted by iterating through the DICOM files located in the configured `DICOM_PATH` directory. The `pydicom` library is used to read each file, providing access to the numerical pixel array and the metadata tags. During this phase, necessary metadata fields are safely extracted using the `safe_get_value` helper function. This function prevents pipeline failures in the event of missing values.

The **transformation phase** focuses on data cleansing, standardization, and the creation of surrogate keys to enforce the star schema structure. Transformation operations are classified into three main areas:

1. Dimensional Key Creation and Management: The `surrogate_key` function generates a deterministic MD5 hash based on dimension attributes, serving as the unique primary key (SK). The `get_or_create` function manages dimension records by checking if an SK already exists in the MongoDB collection before insertion, preventing duplicates.
2. Metadata Standardization and Cleansing: The `format_age` function converts the Patient-Age string (e.g., '062Y') into a clean integer value. The `normalize_pixel_spacing` function maps continuous image quality metrics to predefined quality bins (e.g., 0.6, 0.7, 0.8). The `normalize_contrast_agent` function addresses missing data by standardizing absent or ambiguous ContrastAgent values to the explicit string "No contrast agent."
3. Image Processing: The `dicom_to_jpeg` function processes the raw pixel data. It normalizes the values to the 0-255 range, resizes the image to a standardized 256×256 pixel size, and saves it as a grayscale JPEG file in the `JPEG_PATH` directory. The path to this new file is then stored as an attribute in the central fact table.

The **load phase** manages the connection to MongoDB and the sequential insertion of the transformed data into the dimensional structure. The `run_etl_pipeline` function establishes the connection to the MongoDB instance using the `MONGO_URI` configuration and initializes the client and the target database. The main loading loop, handled by `load_fact_study`, calls dedicated dimension loading functions (e.g., `load_dim_patient`, `load_dim_protocol`) to apply transformations and retrieve the necessary surrogate keys. Once all SKs are retrieved, it collects the remaining fact measures (`exposure_time`, `tube_current`) and inserts a complete record into the central `fact_study` collection.

4 MongoDB

4.1 Collections

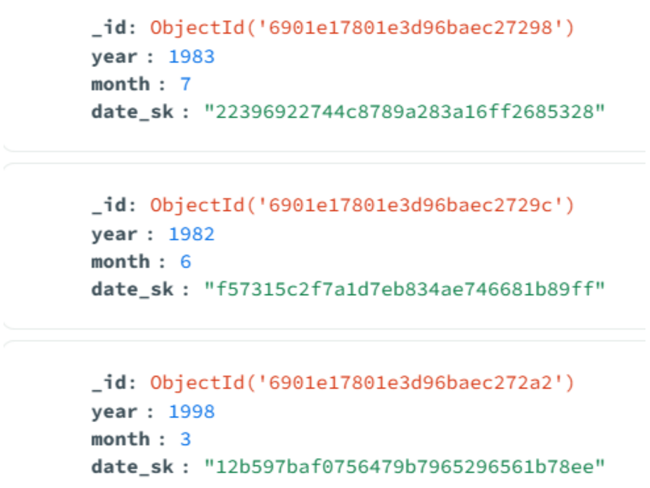
In this section, we will explain briefly each one of the collections created in MongoDB, following the star-schema provided in Figure 1.

Collection "dim_date":

This collection stores data related to date (year and month). This allows our model to filter and group data for specific timestamps.

Collection's structure :

- _id: This is the ObjectId that MongoDB generates automatically for each document.
- date_sk: This is the surrogate key, equivalent to the PK "date_id" in the star-schema shown before, and stores a unique hash key for each date.
- year: The year the study took place.
- month: The month the study took place.



```
_id: ObjectId('6901e17801e3d96baec27298')
year : 1983
month : 7
date_sk : "22396922744c8789a283a16ff2685328"

_id: ObjectId('6901e17801e3d96baec2729c')
year : 1982
month : 6
date_sk : "f57315c2f7a1d7eb834ae746681b89ff"

_id: ObjectId('6901e17801e3d96baec272a2')
year : 1998
month : 3
date_sk : "12b597baf0756479b7965296561b78ee"
```

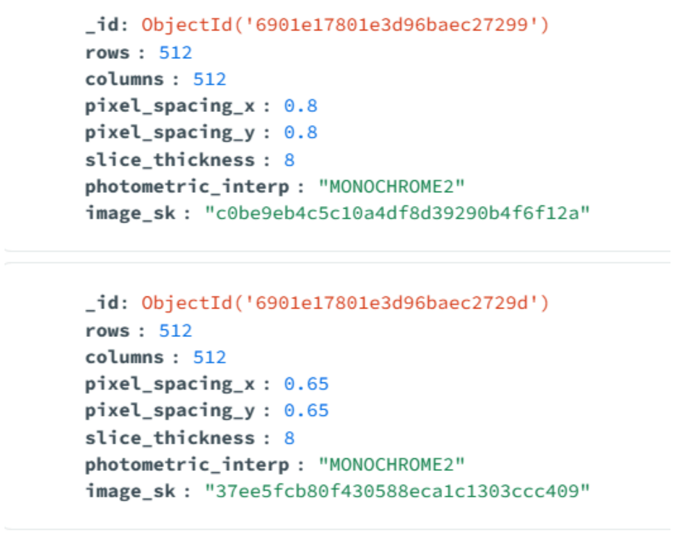
Figure 4: Example of documents from the dim_date collection.

Collection "dim_image":

This collection stores data related to the physical and technical attributes of the medical images. This allows us to filter and group different studies by image properties.

Collection's structure :

- _id: This is the ObjectId that MongoDB generates automatically for each document.
- image_sk: This is the surrogate key, equivalent to the PK "image_id" in the star-schema shown before, and stores a unique hash key for each image.
- rows: The number of rows in the image in pixels.
- columns: The number of columns in the image in pixels.
- pixel_spacing_x: The physical distance from the centers of two adjacent pixels (in mm) with the row direction.
- pixel_spacing_y: The physical distance from the centers of two adjacent pixels (in mm) with the column direction.
- slice_thickness: The thickness of each slide (in mm).
- photometric_interp: This value stores the information about how the pixel data should be interpreted for its visualization.



```
_id: ObjectId('6901e17801e3d96baec27299')
rows : 512
columns : 512
pixel_spacing_x : 0.8
pixel_spacing_y : 0.8
slice_thickness : 8
photometric_interp : "MONOCHROME2"
image_sk : "c0be9eb4c5c10a4df8d39290b4f6f12a"

_id: ObjectId('6901e17801e3d96baec2729d')
rows : 512
columns : 512
pixel_spacing_x : 0.65
pixel_spacing_y : 0.65
slice_thickness : 8
photometric_interp : "MONOCHROME2"
image_sk : "37ee5fcb80f430588eca1c1303ccc409"
```

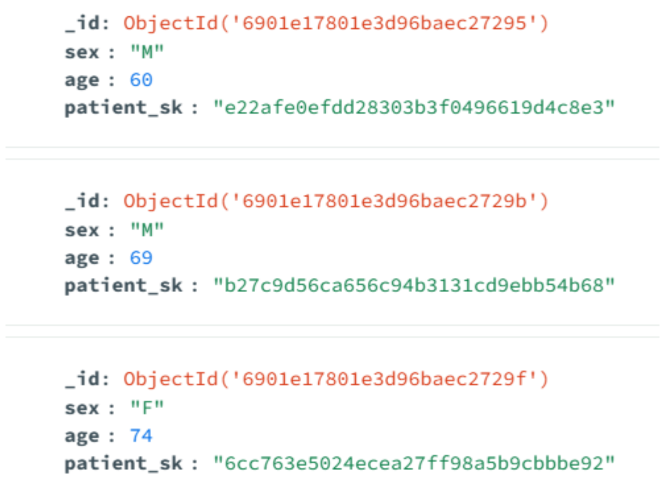
Figure 5: Example of documents from the dim_image collection.

Collection “dim_patient”:

This collection stores information about patients (sex and age). This allows us to filter and group data for these patients’ attributes.

Collection’s structure :

- _id: This is the ObjectId that MongoDB generates automatically for each document.
- patient_sk: This is the surrogate key, equivalent to the PK “patient_id” in the star-schema shown before, and stores a unique hash key for each patient.
- sex: The sex (male or female) of each patient.
- age: The age of each patient.



```
_id: ObjectId('6901e17801e3d96baec27295')
sex : "M"
age : 60
patient_sk : "e22afe0efdd28303b3f0496619d4c8e3"

_id: ObjectId('6901e17801e3d96baec2729b')
sex : "M"
age : 69
patient_sk : "b27c9d56ca656c94b3131cd9ebb54b68"

_id: ObjectId('6901e17801e3d96baec2729f')
sex : "F"
age : 74
patient_sk : "6cc763e5024ecea27ff98a5b9cbbbe92"
```

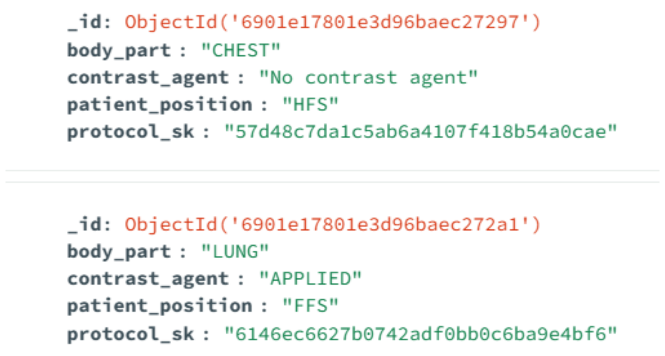
Figure 6: Example of documents from the dim_patient collection.

Collection “dim_protocol”:

This collection stores data about the protocol used for image acquisition (body part, position of the patient,...). This allows our model to filter and group data for different studies.

Collection’s structure :

- _id: This is the ObjectId that MongoDB generates automatically for each document.
- protocol_sk: This is the surrogate key, equivalent to the PK “protocol_id” in the star-schema shown before, and stores a unique hash key for each date.
- body_part: The part of the body studied.
- contrast_agent: Information about whether contrast was applied for the study.
- patient_position: The position of the patient during the study.



```
_id: ObjectId('6901e17801e3d96baec27297')
body_part : "CHEST"
contrast_agent : "No contrast agent"
patient_position : "HFS"
protocol_sk : "57d48c7da1c5ab6a4107f418b54a0cae"

_id: ObjectId('6901e17801e3d96baec272a1')
body_part : "LUNG"
contrast_agent : "APPLIED"
patient_position : "FFS"
protocol_sk : "6146ec6627b0742adf0bb0c6ba9e4bf6"
```

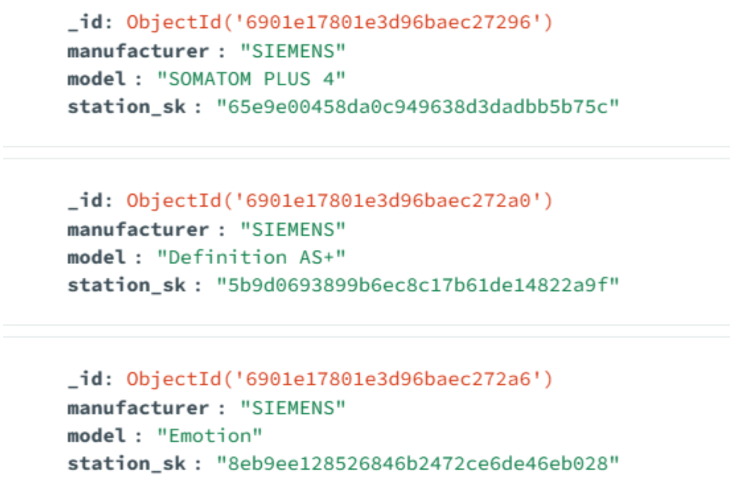
Figure 7: Example of documents from the dim_protocol collection.

Collection "dim_station":

This collection stores data related to the machine (station) used for the study (model and manufacturer). This allows our model to filter and group data by the different machines used.

Collection's structure :

- _id: This is the ObjectId that MongoDB generates automatically for each document.
- station_sk: This is the surrogate key, equivalent to the PK "station_id" in the star-schema shown before, and stores a unique hash key for each date.
- manufacturer: The company that made the machine.
- model: The model of the machine.



```
_id: ObjectId('6901e17801e3d96baec27296')
manufacturer : "SIEMENS"
model : "SOMATOM PLUS 4"
station_sk : "65e9e00458da0c949638d3dadbb5b75c"

_id: ObjectId('6901e17801e3d96baec272a0')
manufacturer : "SIEMENS"
model : "Definition AS+"
station_sk : "5b9d0693899b6ec8c17b61de14822a9f"

_id: ObjectId('6901e17801e3d96baec272a6')
manufacturer : "SIEMENS"
model : "Emotion"
station_sk : "8eb9ee128526846b2472ce6de46eb028"
```

Figure 8: Example of documents from the dim_date collection.

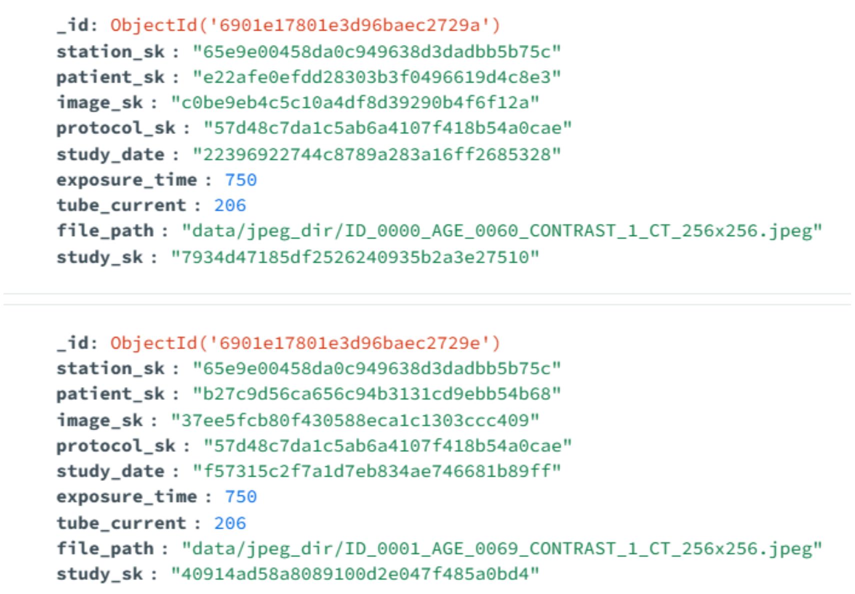
Collection "fact_study":

This collection stores the data equivalent to our fact table in our star schema, connects all dimensions, and stores numeric metrics of the event (study). Each document in this collection represents a unique study event.

Collection's structure :

- _id: This is the ObjectId that MongoDB generates automatically for each document.
- study_sk: This is the surrogate key, equivalent to the PK "study_id" in the star-schema shown before, and stores a unique hash key for each study.
- station_sk: Foreign Key (FK). It is the surrogate key that relates to a document from the "dim_station".
- patient_sk: Foreign Key (FK). It is the surrogate key that relates to a document from the "dim_patient".
- image_sk: Foreign Key (FK). It is the surrogate key that relates to a document from the "dim_image".
- protocol_sk: Foreign Key (FK). It is the surrogate key that relates to a document from the "dim_protocol".
- study_date: Foreign Key (FK). It is the surrogate key that relates to a document from the "dim_date".
- exposure_time: This is a metric fact that stores the exposure time to get the image.
- tube_current: This is a metric fact that stores the current of the tube during the image acquisition.

- file_path: This is a metric fact, unlike the others, it is not used to make aggregation operations or retrieve statistics or metrics. Instead, it stores the new JPEG file path created when running the pipeline.



```
_id: ObjectId('6901e17801e3d96baec2729a')
station_sk : "65e9e00458da0c949638d3dadbb5b75c"
patient_sk : "e22afe0efdd28303b3f0496619d4c8e3"
image_sk : "c0be9eb4c5c10a4df8d39290b4f6f12a"
protocol_sk : "57d48c7da1c5ab6a4107f418b54a0cae"
study_date : "22396922744c8789a283a16ff2685328"
exposure_time : 750
tube_current : 206
file_path : "data/jpeg_dir/ID_0000_AGE_0060_CONTRAST_1_CT_256x256.jpeg"
study_sk : "7934d47185df2526240935b2a3e27510"
```

```
_id: ObjectId('6901e17801e3d96baec2729e')
station_sk : "65e9e00458da0c949638d3dadbb5b75c"
patient_sk : "b27c9d56ca656c94b3131cd9ebb54b68"
image_sk : "37ee5fcb80f430588eca1c1303ccc409"
protocol_sk : "57d48c7da1c5ab6a4107f418b54a0cae"
study_date : "f57315c2f7a1d7eb834ae746681b89ff"
exposure_time : 750
tube_current : 206
file_path : "data/jpeg_dir/ID_0001_AGE_0069_CONTRAST_1_CT_256x256.jpeg"
study_sk : "40914ad58a8089100d2e047f485a0bd4"
```

Figure 9: Example of documents from the fact_study collection.

4.2 MongoDB Queries

As a usage example of our database, we include in this section a set of queries in order to answer some questions, such as:

Query 1: What is the average exposure time and tube current for each manufacturer and model?

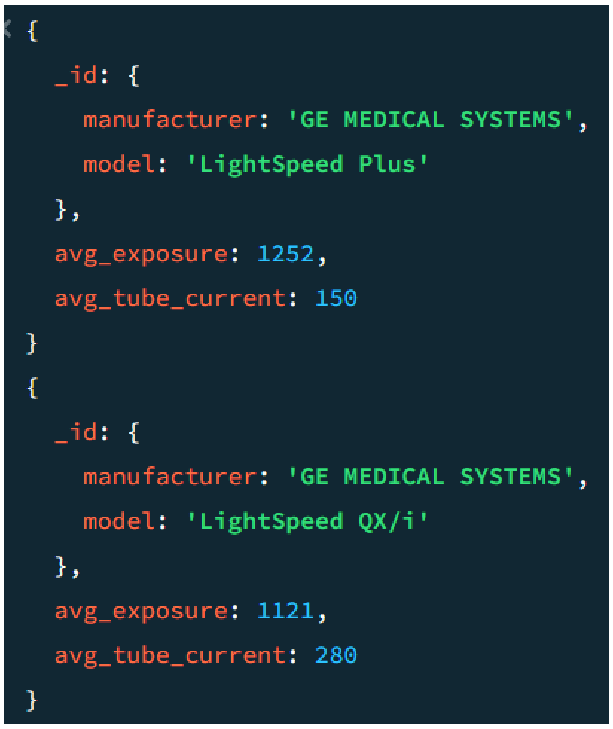
In order to answer this question, we built the following query using the aggregation stages:

1. `$lookup`: First, we join the collection `fact_study` with `dim_station` using the key value `station_sk`, this allows us to retrieve the manufacturer and model of each study.
2. `$unwind`: After that we flatten the array that gives `$lookup`. This gives us a document for each study with its manufacturer and model.
3. `$group`: Then we group all the documents by manufacturer and model, and for each group we make the average `$avg` of exposure time and tube current.
4. `$sort`: Lastly we sort in a descendent way (-1) the averaged exposure times.

```
db.getCollection('fact_study').aggregate(
  [
    {
      $lookup: {
        from: 'dim_station',
        localField: 'station_sk',
        foreignField: 'station_sk',
        as: 'station_info'
      }
    },
    { $unwind: { path: '$station_info' } },
    {
      $group: {
        _id: {
          manufacturer:
            '$station_info.manufacturer',
          model: '$station_info.model'
        },
        avg_exposure: { $avg: '$exposure_time' },
        avg_tube_current: {
          $avg: '$tube_current'
        }
      }
    },
    { $sort: { avg_exposure: -1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

Listing 2: Script for query 1, calculate average exposure time and tube current per scanner model.

As a result of this query, we obtain a list of documents, as shown in the image below:



```
{
  _id: {
    manufacturer: 'GE MEDICAL SYSTEMS',
    model: 'LightSpeed Plus'
  },
  avg_exposure: 1252,
  avg_tube_current: 150
}
{
  _id: {
    manufacturer: 'GE MEDICAL SYSTEMS',
    model: 'LightSpeed QX/i'
  },
  avg_exposure: 1121,
  avg_tube_current: 280
}
```

Figure 10: Example of documents retrieved from the query 1.

Query 2: Which body part requires the longest average exposure time?

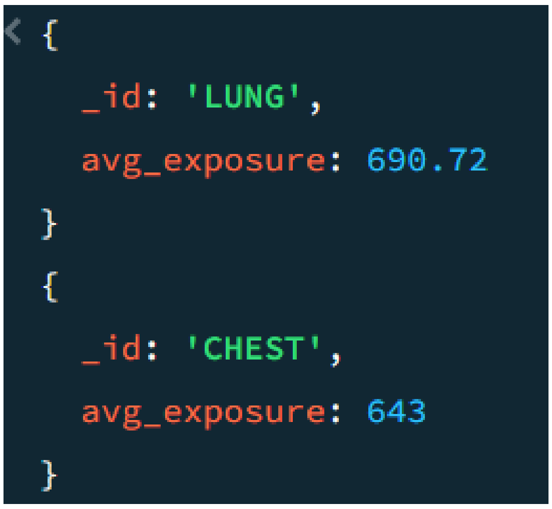
In order to answer this question, we built the following query using the aggregation stages:

1. `$lookup`: First, we join the collection `fact_study` with `dim_protocol` using the key value `protocol_sk`, this allows us to know the body part of each study.
2. `$unwind`: After that we flatten the array that gives `$lookup`. This gives us a document for each study with its body part.
3. `$group`: Then we group all the documents by body part, and for each group we make the average `$avg` of exposure time.
4. `$sort`: Lastly we sort in a descendent way (-1) the averaged exposure times.

```
db.getCollection('fact_study').aggregate(
  [
    {
      $lookup: {
        from: 'dim_protocol',
        localField: 'protocol_sk',
        foreignField: 'protocol_sk',
        as: 'protocol_info'
      }
    },
    { $unwind: { path: '$protocol_info' } },
    {
      $group: {
        _id: '$protocol_info.body_part',
        avg_exposure: { $avg: '$exposure_time' }
      }
    },
    { $sort: { avg_exposure: -1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

Listing 3: Script for query 2, identify the body part with the longest average exposure time.

As a result of this query, we obtain a list of documents, as shown in the image below:



```
{
  _id: 'LUNG',
  avg_exposure: 690.72
}
{
  _id: 'CHEST',
  avg_exposure: 643
}
```

Figure 11: Example of documents retrieved from the query 2.

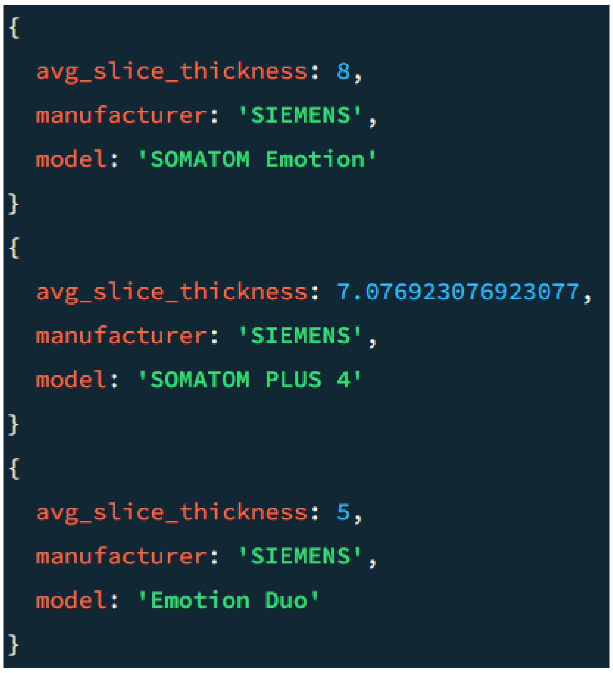
Query 3: What is the variation in average slice thickness across different scanner models for chest studies?

In order to answer this question, we built the following query using the aggregation stages:

1. `$lookup`: First, we join the collection `fact_study ` with `dim_protocol` using the key value `protocol_sk`, this allows us to know the body part of each study.
2. `$unwind`: After that we flatten the array that gives `$lookup`.
3. `$match`: We filter the documents to keep the ones where the body part is "CHEST".
4. `$lookup`: We join the remaining documents with dim_image using the key image_sk.
5. `$unwind`:Flatten the array that gives `$lookup`.
6. `$lookup$`: Another join, this time with dim_station using station_sk.
7. `$unwind`:Flatten the array that gives `$lookup`.
8. `$group`: Group all the documents by manufacturer and model. For each unique group, we calculate the average ($avg) of the slice_thickness.
9. `$sort`. We sort the resulting documents in descending order (-1) by their average slice thickness.
10. `$project `: Optionally, we can modify the output names for clarity.

```
  db.getCollection('fact_study').aggregate(
  [
    {
      $lookup: {
        from: 'dim_protocol',
        localField: 'protocol_sk',
        foreignField: 'protocol_sk',
        as: 'protocol_data'
      }
    },
    { $unwind: { path: '$protocol_data' } },
    {
      $match: {
        'protocol_data.body_part': 'CHEST'
      }
    },
    {
      $lookup: {
        from: 'dim_image',
        localField: 'image_sk',
        foreignField: 'image_sk',
        as: 'image_data'
      }
    },
    { $unwind: { path: '$image_data' } },
    {
      $lookup: {
        from: 'dim_station',
        localField: 'station_sk',
        foreignField: 'station_sk',
        as: 'station_data'
      }
    },
    { $unwind: { path: '$station_data' } },
    {
      $group: {
        _id: {
          manufacturer:
            '$station_data.manufacturer',
          model: '$station_data.model'
        },
        avg_slice_thickness: {
          $avg: '$image_data.slice_thickness'
        }
      }
    },
    {
      $project: {
        _id: 0,
        manufacturer: '$_id.manufacturer',
        model: '$_id.model',
        avg_slice_thickness: 1
      }
    },
    { $sort: { avg_slice_thickness: -1 } }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

Listing 4: Script for query 3, calculate average slice thickness by scanner model for chest studies.



```
{
  avg_slice_thickness: 8,
  manufacturer: 'SIEMENS',
  model: 'SOMATOM Emotion'
}
{
  avg_slice_thickness: 7.076923076923077,
  manufacturer: 'SIEMENS',
  model: 'SOMATOM PLUS 4'
}
{
  avg_slice_thickness: 5,
  manufacturer: 'SIEMENS',
  model: 'Emotion Duo'
}
```

Figure 12: Example of documents retrieved from the query 3.

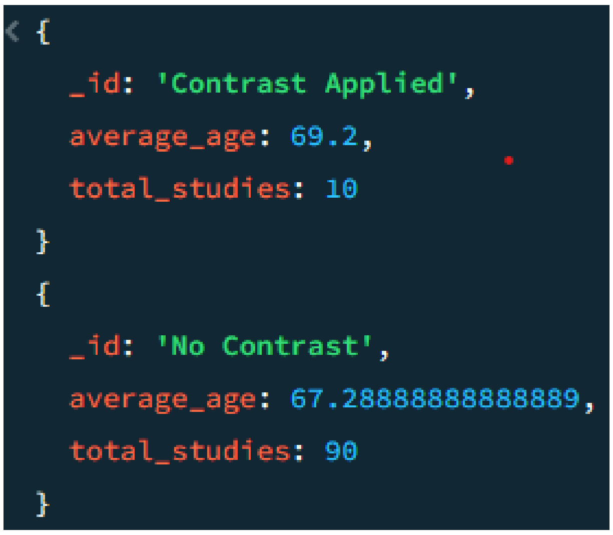
Query 4: How does the average age differ between patients requiring contrast and those who do not?

In order to answer this question, we built the following query using the aggregation stages:

1. `$lookup`: First, we join the collection `fact_study` with `dim_patient` using the key value `patient_sk`.
2. `$unwind`: After that we flatten the array that gives `$lookup`.
3. `$lookup`: We make another join with dim_protocol using the key protocol_sk.
4. `$unwind`:Flatten the array that gives `$lookup`.
5. `$addFields`: As our data is not boolean for contrast agent, and varies among the different documents, we create a new field called contrast_group. We use a `$cond` to separate the values into “No Contrast” and “Contrast Applied”.
6. `$group`: Group all the documents by the new field contrast group. For each group, we calculate the average ($avg) age of the patients and count the total studies using `$sum:1`.

```
  db.getCollection('fact_study').aggregate(
  [
    {
      $lookup: {
        from: 'dim_patient',
        localField: 'patient_sk',
        foreignField: 'patient_sk',
        as: 'patient_data'
      }
    },
    { $unwind: { path: '$patient_data' } },
    {
      $lookup: {
        from: 'dim_protocol',
        localField: 'protocol_sk',
        foreignField: 'protocol_sk',
        as: 'protocol_data'
      }
    },
    { $unwind: { path: '$protocol_data' } },
    {
      $addFields: {
        contrast_group: {
          $cond: {
            if: {
              $eq: [
                '$protocol_data.contrast_agent',
                'No contrast agent'
              ]
            },
            then: 'No Contrast',
            else: 'Contrast Applied'
          }
        }
      }
    },
    {
      $group: {
        _id: '$contrast_group',
        average_age: {
          $avg: '$patient_data.age'
        },
        total_studies: { $sum: 1 }
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

Listing 5: Script for query 4, compare average patient age for studies with and without contrast media.



```
< {
    _id: 'Contrast Applied',
    average_age: 69.2,
    total_studies: 10
  }
  {
    _id: 'No Contrast',
    average_age: 67.28888888888889,
    total_studies: 90
  }
```

Figure 13: Example of documents retrieved from the query 4.

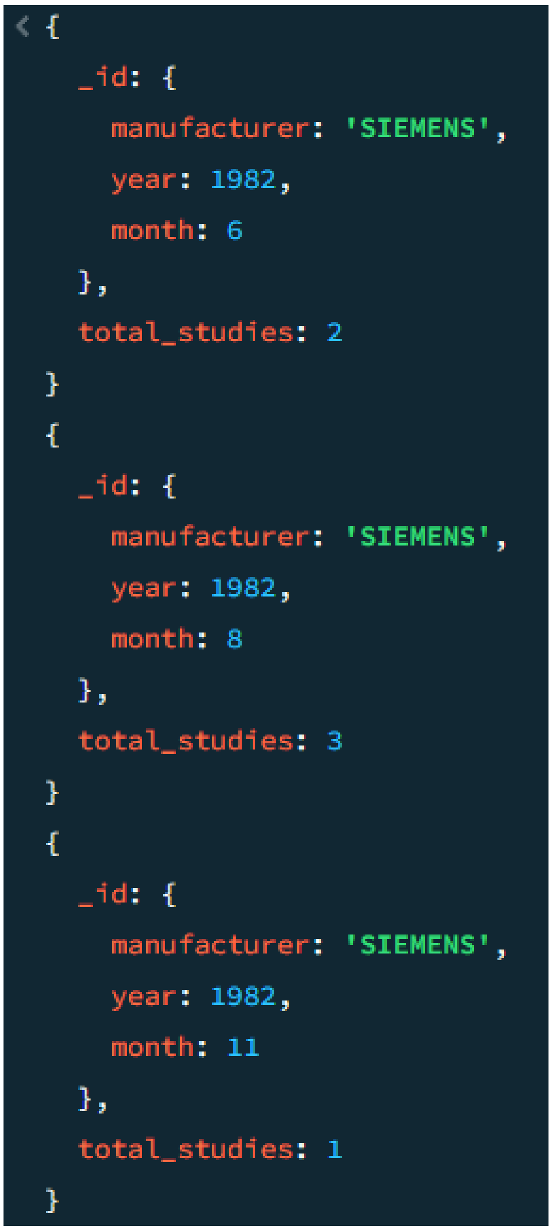
Query 5: How has the volume of studies changed over time for each equipment manufacturer?

In order to answer this question, we built the following query using the aggregation stages:

1. `$lookup`: First, we join the collection `fact_study` with `dim_station` using the key value `station_sk`.
2. `$unwind`: After that we flatten the array that gives `$lookup`.
3. `$lookup`: We make another join, this time with dim_date, we connect the study_date field in fact_study with the date_sk field in dim_date. This gives us the year and month the study took place.
4. `$unwind`:Flatten the array that gives `$lookup`.
5. `$group`: Group all the documents using a composite key of manufacturer, year, and month. For each group, we count the total_studies using `$sum:1`.
6. `$sort`: Finally, we sort in ascending order (1) first by the year and then by month.

```
db.getCollection('fact_study').aggregate(
  [
    {
      $lookup: {
        from: 'dim_station',
        localField: 'station_sk',
        foreignField: 'station_sk',
        as: 'station_data'
      }
    },
    { $unwind: { path: '$station_data' } },
    {
      $lookup: {
        from: 'dim_date',
        localField: 'study_date',
        foreignField: 'date_sk',
        as: 'date_data'
      }
    },
    { $unwind: { path: '$date_data' } },
    {
      $group: {

        _id: {

          manufacturer: '$station_data.manufacturer',

          year: '$date_data.year',

          month: '$date_data.month'

        },

        total_studies: { $sum: 1 }
      }
    },
    {
      $sort: {
        '_id.year': 1,
        '_id.month': 1
      }
    }
  ],
  { maxTimeMS: 60000, allowDiskUse: true }
);
```

Listing 6: Script for query 5, count total studies over time grouped by equipment manufacturer.



```
< {
    _id: {
      manufacturer: 'SIEMENS',
      year: 1982,
      month: 6
    },
    total_studies: 2
  }
  {
    _id: {
      manufacturer: 'SIEMENS',
      year: 1982,
      month: 8
    },
    total_studies: 3
  }
  {
    _id: {
      manufacturer: 'SIEMENS',
      year: 1982,
      month: 11
    },
    total_studies: 1
  }
```

Figure 14: Example of documents retrieved from the query 5.

Bibliography

[1] B. Albertina *et al.*, "The Cancer Genome Atlas Lung Adenocarcinoma Collection (TCGA-LUAD)." [Online]. Available: https://www.cancerimagingarchive.net/collection/tcga-luad/