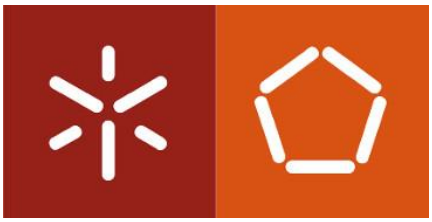


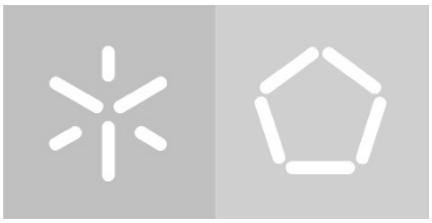
Universidade do Minho
Escola de Engenharia
Departamento de Informática

Carolina Cunha, A80142
Hugo Faria, A81283
João Diogo Mota, A80791
Rodolfo Silva, A81716

Casuística do Serviço de Urgência (janeiro a outubro de 2018)

Grupo 8

Janeiro 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Carolina Cunha, A80142
Hugo Faria, A81283
João Diogo Mota, A80791
Rodolfo Silva, A81716

Casuística do Serviço de Urgência (janeiro a outubro de 2018)

Grupo 8

Aplicações Informáticas na Biomedicina
Mestrado Integrado em Engenharia Informática

Trabalho efetuado sob orientação de
José Machado
Daniela Oliveira

Janeiro 2021

RESUMO

O presente projeto consistiu no armazenamento e análise de informações clínicas, de forma a permitir auxiliar no suporte à decisão clínica. Para este efeito, foi realizado, numa primeira instância, um tratamento de dados que permitisse eliminar a sua inconsistência e retificar quaisquer valores nulos existentes. Subsequentemente, foi realizada a modelação lógica do *Data Warehouse*, tendo por base os dados a analisar. Sobre esta modelação seguiu-se o processo de carregamento dos dados (recorrendo a *scripts* SQL e à plataforma *Talend)*, inserindo-os no *Data Warehouse* implementado. Posteriormente, procedeu-se à implementação de uma área de retenção para futuras atualizações deste *Data Warehouse*, das formas incremental e total. Numa segunda fase, foram analisados possíveis indicadores importantes à decisão clínica, através da plataforma *Tableau Desktop*, entrando em discussão com alguns profissionais de saúde, de forma a validar a importância dos mesmos. Finalmente, implementou-se um modelo de uma possível aplicação web para visualização e estudo dos diversos indicadores observados, com recurso à *framework* Vue.js.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ANEXOS

LISTA DE SIGLAS E ACRÓNIMOS

ARS	Administração Regional de Saúde
BI	*Business Intelligence*
CE	Consulta Externa
ECA	Estudo de Condutividade Auditiva
ECG	Eletrocardiograma
EN	Escala Numérica
ETL	*Extract-Transform-Load*
IBP	Inibidores da Bomba de Protões
ICD	*International Classification of Diseases*
IM	Intramuscular
IV	Intravenosa
MCDT	Meios Complementares de Diagnóstico e Terapêutica
PvP	Preço de Venda ao Público
RM	Ressonância Magnética
RMN	Ressonância Magnética Nuclear
SNS	Serviço Nacional de Saúde
SQL	*Structured Query Language*
STM	Sistema de Triagem de Manchester
SU	Serviço de Urgência
TAC	Tomografia Axial Computarizada

1

INTRODUÇÃO

Atualmente, a tecnologia tem uma importância essencial em todas as vertentes da sociedade, (como por exemplo nos ramo empresarial, de entretenimento ou de saúde). Nesto ramo da saúde, é cada vez mais habitual e indispensável o uso da tecnologia, destacando-se as Bases de Dados para manter todo o tipo de registos, como o registo clínico dos pacientes (essencial para a monitorização do seu estado de saúde). As funcionalidades destas Bases de Dados não se esgotam no armazenamento dos registos de diversos assuntos; servem também como base de informação consolidada e tratada para ser utilizada em processos de análise de dados. Esta análise é fundamental no planeamento e desenvolviemnto, sendo esta área particularmente importante, tendo em conta que a melhoria do funcionamento do sistema implicará melhor assistência ao ser humano doente.

A unidade curricular de Aplicações Informáticas na Biomedicina pretende transmitir aos alunos a importância do armazenamento e análise de informações clínicas. Para isso, foi fornecido um *dataset* contendo diversos dados relacionados com episódios de urgência, com o intuito de se analisar e construir um *Data Warehouse* com a informação devidamente selecionada e tratada segundo os processos ETL. Por fim, foram criados indicadores que permitiram analisar aspetos que o grupo considerou relevantes.

2

PRELIMINARES

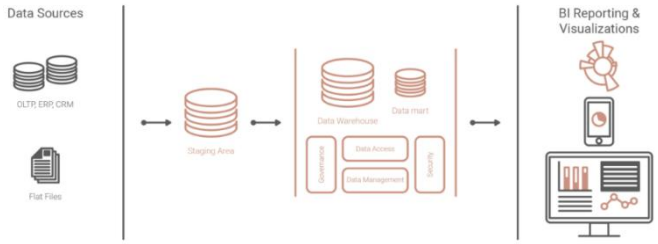
Previamente à realização deste projeto, foram estudados diferentes pressupostos imprescindíveis ao suporte à decisão clínica.

Assim sendo, apresenta-se a noção de *Data Warehouse*, uma base de dados de suporte à decisão que é mantida separadamente da base operacional da organização. É um sistema de agregação de dados derivados de diferentes fontes de informação, responsável pela correlação entre estes.

As características proeminentes de um *Data Warehouse* são:

- É **orientado à modelação** e análise de dados para a tomada de decisões;
- Fornece uma visão simples e concisa sobre questões de um tema particular, através da exclusão de dados que não são importantes no suporte ao processo de decisão;
- É **integrado**, uma vez que é constituído por integração de múltiplas e heterogéneas fontes de dados;
- É **consistente** nas nomenclaturas, formatos e codificação de dados;
- É **invariante no tempo**, pelo que os dados inseridos não sofrem qualquer mutação, estando a estes associado um determinado período;
- Não é **volátil**, isto é, os dados não são apagados aquando da inserção de novos dados. Permite apenas operações de *loading* e *access*.

Figura 1 | Arquitetura de um *Data Warehouse*



A **modelação (dimensional)** é uma técnica de estrutura de dados otimizada para armazenamento de dados num *Data Warehouse*. Este modelo tem como objetivo otimizar a base de dados para uma obtenção rápida dos dados. A modelação de um *Data Warehouse* pressupõe a noção de **dimensões**, **factos** e **variáveis**. Um **facto** representa um item, transação ou evento de um negócio e é utilizado para analisar o processo de negócio de uma organização. É representado por valores numéricos e implementado em tabelas de factos. As **dimensões** são as possíveis formas de visualizar os dados. Estas descrevem e classificam os elementos que

participam num facto. Por último, as **variáveis** são os atributos numéricos que representam um facto [1].

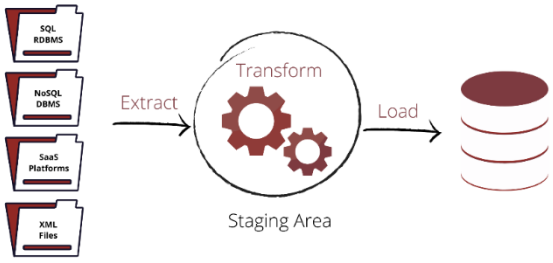
A modelação dimensional pode ser implementada de acordo com os diferentes tipos de esquemas existentes.

- **Esquema em Estrela** - É caracterizado pela presença de uma tabela de factos no centro da estrela, com um número variável de tabelas de dimensões. Neste tipo de esquema, existe uma única hierarquia de relações.
- **Esquema em Floco de Neve** - Este esquema é uma extensão do esquema em Estrela, e adiciona dimensões extra. As tabelas de dimensão são normalizadas, o que divide a informação por diversas tabelas. Para este tipo de esquema, encontram-se múltiplas hierarquias.
- **Esquema de Constelação de Factos** – Caracteriza-se pela presença de duas ou mais tabelas de factos que partilham tabelas de dimensão entre si. É visto como uma coleção de esquemas em Estrela.

O sistema ***Extract-Transform-Load*** (ETL) é a base do *Data Warehouse*. Trata-se de um processo para extrair dados de um sistema de Bases de Dados, sendo esses dados processados, modificados, e posteriormente inseridos numa outra base de dados [2]. A utilização deste sistema apresenta vantagens tais como:

- Facilita a **análise** e **criação** de relatórios sobre dados relevantes;
- Fornece garantias de **qualidade** dos dados;
- Garante uma maior **eficiência** e menos gasto de recursos durante todo o seu processo;
- Fornece **facilidade** de codificação.

Figura 2 | Sistema ETL



O *Business Intelligence* (BI) combina a análise de negócios, mineração de dados, visualização de dados, ferramentas e infraestruturas de dados para auxiliar as empresas nas tomadas de decisões baseadas em dados [3].

3

TRATAMENTO DOS DADOS

Os dados fornecidos para análise pertencem a um conjunto de dados real repartido por múltiplos ficheiros. A informação nestes contida dizem respeito à **casuística do Serviço de Urgência dos meses de janeiro a outubro de 2018** de um centro hospitalar.

A primeira etapa no processo de ETL é a extração dos dados do sistema de origem para uma área de retenção (*schema 'dados'* referido na página 9) [4].

Assim, juntamente com alguns profissionais de saúde, foram analisados individualmente os diversos componentes do *dataset*, e isoladas as informações relevantes para o suporte à decisão clínica.

A segunda fase deste processo compreende a transformação dos dados, melhorando a integridade dos mesmos. Esta transformação é composta por vários sub-processos, entre os quais [4]:

- Processo de resolução de dados com valores inconsistentes e nulos;
- Regra de formatação a aplicar ao conjunto de dados;
- Exclusão dos dados redundantes;
- Remoção dos dados inutilizáveis e sinalização das anomalias;
- Organização dos dados de acordo com o tipo.

São descritas, em seguida, as decisões tomadas para tratamento dos dados.

Para as intervenções clínicas, foram eliminadas todas as notas de intervenção e de cancelamento, uma vez que apresentavam poucos dados e estes eram inconsistentes (visto serem escritos manualmente pelos profissionais de saúde). Foram, ainda, manipulados os dados relativos às datas e identificador do profissional de cancelamento de uma intervenção. Uma vez que eram escassas as ocorrências de cancelamento de uma intervenção, era frequente a existência de valores nulos. Sabendo que um *Data Warehouse* não pode conter qualquer valor nulo, as datas de cancelamento foram convertidas para um valor binário (onde 1 representa o cancelamento de uma intervenção e 0 o contrário), e ao identificador dos respetivos profissionais responsáveis pelo cancelamento desta intervenção foi atribuído o valor -1.

No que diz respeito à descrição de uma intervenção clínica, foi sugerida a junção dos diversos procedimentos hospitalares. Na Tabela 1, encontram-se os diferentes procedimentos considerados.

Tabela 1 | Procedimentos hospitalares

Procedimentos	Procedimentos - detalhado
Terapêutica Intravenosa	Injeções por Via IV, Soroterapia
Terapêutica Subcutânea	Injeções por Via Subcutânea
Intramuscular	Injeções por Via IM
Inalado	Oxigenoterapia/Inaloterapia/Ventiloterapia
Intervenção Diagnóstica	ECG, Endoscopia, Colheita de Urina, entre outros
Intervenção Terapêutica	Imobilização, Penso, Hemodialise, entre outros
Outros	Ventilação e doenças (Hemofilia)

IV – Intravenoso, IM – Intramuscular, ECG – Eletrocardiograma

Nas prescrições, foram removidos os valores de PvP (Preço de Venda ao Público) e de comparticipação, uma vez que só apresentavam o valor zero. A posologia foi também eliminada, dada a inconsistência dos dados. Para as descrições dos fármacos prescritos, estas foram agrupadas de acordo com o grupo farmacológico do respetivo fármaco, permitindo uma redução considerável do número distinto de fármacos presentes no conjunto de dados. Desta forma, foi possível utilizar esta informação para posterior análise. A Tabela 2 apresenta os grupos farmacológicos mais comuns no Serviço de Urgência (SU), de acordo com o grupo farmacológico.

Tabela 2 | Grupos farmacológicos mais comuns no SU

Analgésico	Colirios
Anti Prostatico	Corticóides
Antiagregante Plaquetar	Dispositivos Externos
Anti Arrítmicos	Diurético
Antibiótico	Gota
Anticoagulantes	IBP
Anti Diabéticos	Imunomoduladores
Anti Epiléticos	Laxantes / Antidiarreicos
Antifibrinolítico	Outros
Anti Hipertensores	Pro-Cinéticos
Anti Histamínico	Psicotrópico
Anti Isquémicos	Tiróideus
Anti Vertiginosos	Vitaminas/ferro/iões/suplementos
Broncodilatadores	nutricionais

***IBP* - Inibidores da Bomba de Protões**

Por fim, removeu-se as notas de diagnóstico (pela presença de dados inconsistentes) das informações relativas aos episódios de urgência. Devido à existência de algumas datas de alta sem valores atribuídos, estas foram manipuladas, de forma a permitir a sua inserção do *Data Warehouse* (atribuiu-se-lhes o valor 1111-11-11 00:00:00).

A codificação clínica hospitalar traduz-se na codificação, mediante nomenclaturas e sistemas de codificação adequados, dos procedimentos, diagnósticos e atos que caracterizam o contacto do utente com o hospital (seja em internamento, hospital de dia, consulta externas ou urgências) [5].

O *International Classification of Diseases* (ICD) é o sistema oficial de atribuição de códigos de diagnósticos para classificação e codificação da informação de morbilidade e mortalidade para fins estatísticos e indexação dos registos hospitalares por doença e intervenções cirúrgicas. Em Portugal, é utilizado para efeitos de codificação clínica das altas hospitalares (de

internamento e parte do ambulatório) possibilitando, assim, a caracterização sistematizada da morbilidade hospitalar [6].

Para uma análise mais objetiva, optou-se por observar unicamente o nível um da hierarquia (nível mais abrangente) dos diferentes códigos de diagnóstico (ICD-9). Assim, é possível analisar as principais patologias observadas no SU em Portugal. Na Tabela 3, encontram-se discriminados os códigos de diagnóstico e respetivas classificações para o primeiro nível da hierarquia.

Tabela 3 | Códigos de diagnóstico e respetivas classificações para o primeiro nível da hierarquia (ICD-9)

Códigos de Diagnóstico	Classificação
001-139	Infectious and Parasitic Diseases
140-239	Neoplasms
240-279	Endocrine, Nutritional and Metabolic Diseases, and Immunity Disorders
280-289	Diseases of the Blood and Blood-forming Organs
290-319	Mental Disorders
320-389	Diseases of the Nervous System and Sense Organs
390-459	Diseases of the Circulatory System
460-519	Diseases of the Respiratory System
520-579	Diseases of the Digestive System
580-629	Diseases of the Genitourinary System
630-679	Complications of Pregnancy, Childbirth and the Puerperium
680-709	Diseases of the Skin and Subcutaneous Tissue
710-739	Diseases of the Musculoskeletal System and Connective Tissue
740-759	Congenital Anomalies
760-779	Certain Conditions Originating in the Perinatal Period
780-799	Symptoms, Signs and Ill-defined Conditions
800-999	Injury and Poisoning
V	Supplementary Classification of Factors Influencing Health Status and Contact with Health Services
E	Supplementary Classification of External Causes of Injury and Poisoning

Foi criado um *script* em *python* para resolver a inconsistência de algumas datas presentes nos ficheiros *.csv* (Anexo 1). Para além disto, este permitiu a correção de linhas com valores nulos no ficheiro correspondente aos exames realizados. Foi criado um outro *script*, também em *python*, que garante que todos os episódios de urgência tenham a si associados um exame, prescrição e intervenção, com os a indicação de que estes não foram realizados (Anexo 2).

4

MODELAÇÃO LÓGICA

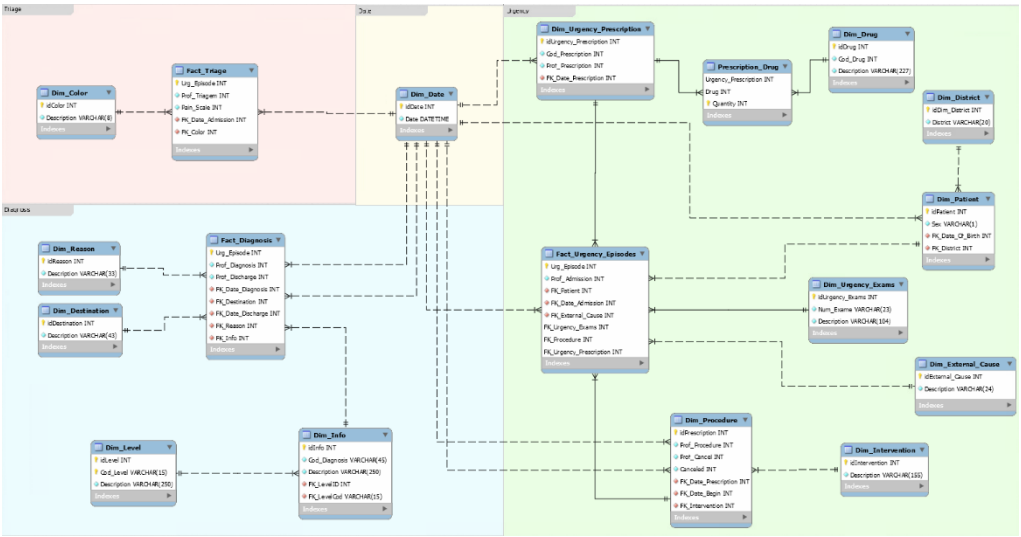
A modelação lógica é uma etapa essencial em qualquer sistema de Base de Dados relacional. Esta etapa permite representar uma Base de Dados como uma série de relacionamentos como forma de armazenar e processar os dados desta. O modelo relacional tem diversas vantagens, tais como permitir um elevado grau de independência de dados; permitir uma vista mais simplificada da estrutura da Base de Dados; diminuir a complexidade do trabalho de um administrador de Bases de Dados; eliminar problemas de redundância, consistência, entre outros.

A modelação lógica teve como início a definição das dimensões e factos, bem como das variáveis que lhes correspondem.

Uma vez que se decidiu realizar uma análise generalizada dos dados, houve a necessidade de trabalhar sobre a sua totalidade. Assim, optou-se pela utilização de um esquema de Constelação de Factos, sendo definidas várias dimensões e as seguintes relações de factos:

- **Fact_Urgency_Episodes** - Dados centrados nos episódios de urgências;
- **Fact_Triage** - Dados centrados no processo de triagem de um episódio;
- **Fact_Diagnosis** - Dados centrados em todo o processo de diagnóstico de um paciente.

Figura 3 | Modelo Lógico



No modelo apresentado Figura 3, é possível distinguir as três relações de factos criadas, bem como vários níveis de hierarquias de dimensões. Deste modelo, retiram-se alguns apontamentos importantes de salientar:

- Todas as datas encontram-se agrupadas numa relação *Dim_Date*, sendo esta partilhada por várias relações, bem como pelas três tabelas de factos existentes;

- Na relação *Dim_Level*, apresenta-se um identificador do nível, o código no nível e a sua descrição, de forma a ser possível identificar todos os códigos de diagnóstico de todos os níveis presentes no *dataset;*

- Na relação *Dim_Info*, encontram-se as informações relativas à classificação do código de diagnóstico segundo o ICD-9.

- Foi necessária a utilização de um relacionamento N:N (relacionamento que gera uma relação extra, contendo as chaves primárias de ambas as tabelas) entre as relações *Dim_Urgency_Prescription* e *Dim_Drug*, de forma a ser possível um medicamento estar presente em várias prescrições, bem como uma prescrição conter vários medicamentos. Dada a necessidade de se representar a quantidade do medicamento prescrito, foi adicionado o atributo que representa essa quantidade da tabela originária do relacionamento N:N;

- Em algumas relações, como por exemplo *Dim_Drug* ou *Dim_District*, foi adicionado um atributo extra, de forma a conter uma chave primária, auto incremental, para identificação de cada uma dessas relações.

5

POVOAMENTO DAS RELAÇÕES

Para o povoamento das relações implementadas, foi criado um *schema 'dados'* para onde foram importados os dados dos ficheiros do *dataset*. Para este fim, recorreu-se à funcionalidade *Table Data Import Wizard* do *MySQL Workbench*.

O último passo no processo de ETL é o carregamento dos dados para o seu destino. Estes dados podem ser carregados todos de uma vez (*full load*) ou em intervalos agendados (*incremental load*) [4]. O povoamento inicial do *Data Warehouse* foi realizado de acordo com o cenário *full load*.

5.1. Tabelas de dimensão

O processo de povoamento das tabelas de dimensão divide-se na procura de todos os valores distintos de um campo a adicionar ao *Data Warehouse* e posterior inserção no mesmo.

Na Figura 4, encontra-se um exemplo de povoamento para as relações mais elementares. Para este tipo de relação, apenas é necessário extrair os valores diretamente da tabela para onde foi importado o ficheiro respetivo, no *schema 'dados'*.

Figura 4 | Povoamento da Tabela Dim_Color

```
INSERT INTO Dim_Color
SELECT DISTINCT ID_COLOR, DESC_COLOR FROM `dados`.`urgency_episodes`;
```

Relativamente às relações mais complexas, isto é, que têm como chave estrangeira a chave primária de outra relação, foi necessário atuar sobre o relacionamento entre estas. Assim sendo, para cada inserção, é validada a existência de um dado nas restantes relações, obtendo a sua chave primária.

Figura 5 | Povoamento da Tabela Dim_Patient

```
INSERT INTO Dim_Patient (Sex, FK_Date_Of_Birth, FK_District)
SELECT uE.SEX, d.idDate, di.idDistrict
FROM `dados`.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date = uE.DATE_OF_BIRTH
LEFT JOIN Dim_District di ON di.District = uE.DISTRICT;
```

Para relações com um elevado número de operações, foi necessário dividir as transações em subconjuntos mais reduzidos de dados, como foi o caso da *Dim_Urgency_Prescription*.

Figura 6 | Povoamento da Tabela Dim_Urgency_Prescription

```
INSERT INTO Dim_Urgency_Prescription (Cod_Prescription, Prof_Prescription,
FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION, d.idDate
FROM `dados`.`urgency_prescriptions` uP
LEFT JOIN Dim_Date d ON d.Date = STR_TO_DATE(uP.DT_PRESCRIPTION,"%Y/%m/%d %T") WHERE
'COD_PRESCRIPTION'>=16369810 AND 'COD_PRESCRIPTION'>=16420000
```

5.2. Tabelas de factos

O povoamento das tabelas de factos é realizado à posteriori do povoamento da totalidade das tabelas de dimensão, uma vez que utiliza valores contidos nas últimas.

Para cada relação, foi realizado um conjunto de uniões que permitam relacionar cada linha da tabela do *dataset* a uma entrada na tabela de factos, associando esta às chaves primárias das tabelas de dimensão.

Na Figura 7, observa-se o povoamento da tabela de factos *Fact_Diagnosis*. Uma vez que o atributo *Urg_Episode* é um identificador único, pode corresponder à chave primária de cada entrada na tabela. Para além disto, destaca-se a ligação à tabela *Dim_Date* duas vezes, correspondendo às datas de diagnóstico e alta, respetivamente.

Figura 7 | Povoamento da Tabela Fact_Diagnosis

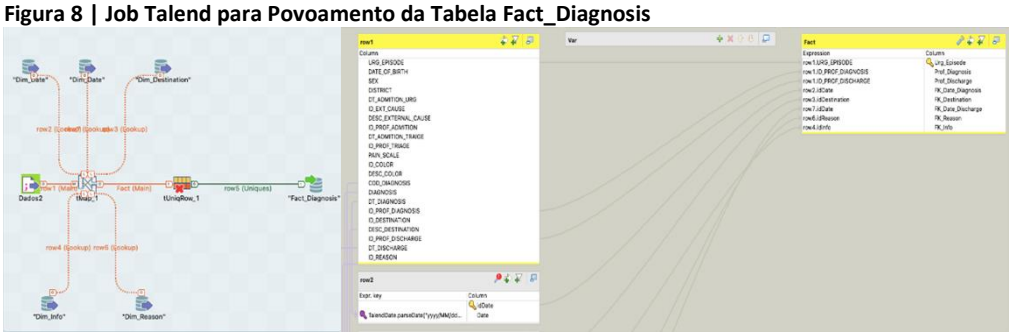
```
INSERT INTO Fact_Diagnosis (Urg_Episode, Prof_Diagnosis, Prof_Discharge,
FK_Date_Diagnosis, FK_Destination, FK_Date_Discharge, FK_Reason,FK_Info)
SELECT DISTINCT a1.URG_EPISODE, a1.ID_PROF_DIAGNOSIS, a1.ID_PROF_DISCHARGE,
a2.idDate,a3.idDestination,a4.idDate,a5.idReason,a6.idInfo
FROM `dados`.`urgency_episodes` a1
INNER JOIN Dim_Date a2
ON STR_TO_DATE(a1.DT_DIAGNOSIS,"%Y/%m/%d %T") = a2.Date
INNER JOIN Dim_Destination a3 ON a1.DESC_DESTINATION = a3.Description
INNER JOIN Dim_Date a4 ON STR_TO_DATE(a1.DT_DISCHARGE,"%Y/%m/%d %T") = a4.Date
INNER JOIN Dim_Reason a5 ON a1.DESC_REASON = a5.Description
INNER JOIN Dim_Info a6 ON a1.COD_DIAGNOSIS = a6.Cod_Diagnosis
AND a1.DIAGNOSIS = a6.Description;
```

O Anexo 3 contém todo o povoamento realizado através de um *script* SQL.

5.3. Povoamento com recurso ao Talend

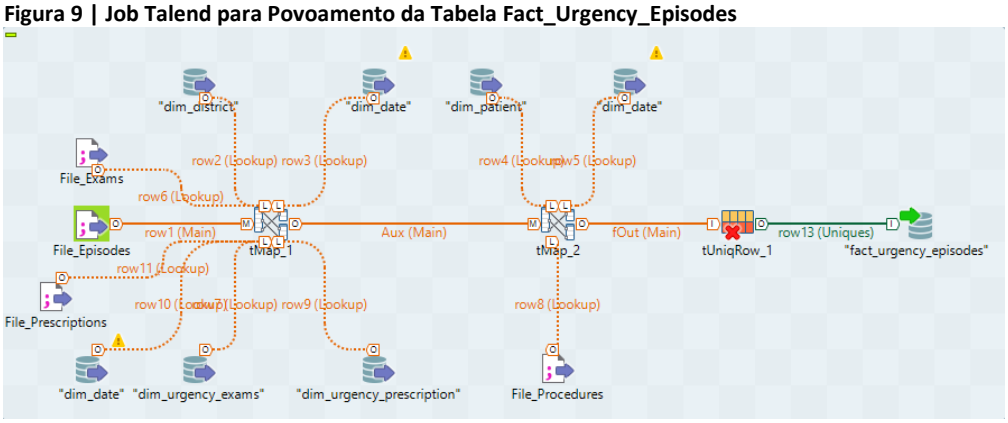
Recorreu-se, ainda, à plataforma de ETL para integração de dados, *Talend*. Através da conexão à Base de Dados, é selecionado um ficheiro *input* (tabela povoada a partir do *dataset*) e cinco relações input, sobre as quais é efetuado um mapeamento. Este mapeamento seleciona os dados necessários ao preenchimento da tabela de factos *Fact_Diagnosis*. Em seguida, os dados são filtrados e inseridos de forma distinta na tabela de factos (Figura 8).

Figura 8 | Job Talend para Povoamento da Tabela Fact_Diagnosis



A tabela de factos *Fact_Urgency_Episodes* tem uma chave primária composta pelo episódio de urgência, identificador do procedimento realizado, identificador da prescrição e identificador da intervenção realizada. Por este motivo, foi necessária a recolha de informação dos diversos ficheiros do conjunto de dados (Figura 9). No Anexo 4, encontram-se os restantes processos de povoamento do *Data Warehouse* através do *Talend*.

Figura 9 | Job Talend para Povoamento da Tabela Fact_Urgency_Episodes



5.4. Comparação dos métodos utilizados

A utilização do *Talend* permite um povoamento eficiente das relações pertencentes ao modelo lógico, possibilitando o encadeamento de ações complexas com um número reduzido de parâmetros.

No entanto, o seu manuseamento revela-se mais complexo e demorado do que o realizado através de *scripts* SQL.

6

ATUALIZAÇÃO DO DATAWAREHOUSE

Um dos pontos essencial num *Data Warehouse* é a existência de mecanismos que permitam a atualização incremental e/ou diferencial dos dados.

Com o objetivo de abordar os dois tipos de atualização de dados propostos e explorar as suas vantagens e desvantagens, foram criadas diversas funções e procedimentos em SQL, de forma a serem construídos os dois métodos de atualização seguindo a metodologia mais favorável.

Após uma pesquisa, conclui-se que atualmente tem sido abandonada a ideia de carregar um grande número de dados de uma vez, sendo o processo de tratamento e inserção de dados no *Data Warehouse* realizada a cada escrita da aplicação.

6.1 Atualização Incremental dos dados

A atualização de dados incremental carateriza-se pela adição dos dados não existentes previamente no *Data Warehouse*. Este método permite que a importação dos dados não seja realizado todo de novo, permitindo poupar recursos e tempo. Por outro lado, este processo torna-se mais complexo por serem necessárias verificações constantes de forma a evitar inconsistências e repetições no *Data Warehouse*. Outro problema com este tipo de atualização é que, ao contrário de uma atualização diferencial, se ocorrer algum erro, não é possível fazer todo o carregamento de dados novamente.

A solução apresentada para a atualização dos dados de forma incremental centrou-se na criação de um novo *schema*, replicando o do *Data Warehouse,* com o objetivo de este armazenar os novos dados que se pretende inserir no *Data Warehouse*, após serem tratados. Deste modo, são eliminadas possíveis incongruências durante o processo de *Business Intelligence* permitindo, apenas quando pretendido, a adição dos novos dados no *Data Warehouse*.

A Figura 10 apresenta a metodologia adotada para inserção incremental de dados, utilizando o exemplo da inserção de uma nova triagem. No exemplo apresentado, bem como em todos os procedimentos criados para este efeito, é essencial verificar se algum valor é nulo e, em caso afirmativo, tratar corretamente estes valores de forma a serem inseridos na área de retenção.

Figura 10 | Exemplo de Inserção na Área de Retenção

```
DROP PROCEDURE IF EXISTS InserirNovaTriagem;
DELIMITER $$
CREATE PROCEDURE InserirNovaTriagem(UEpisode INT, TProfissional INT, PainScale INT,
DataAdmissao VARCHAR(100), idCor INT, Cor VARCHAR(8))
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE nn_TProfissional INT;
    DECLARE nn_PainScale INT;
    DECLARE nn_DataAdmissao VARCHAR(100);
    DECLARE nn_idCor INT; DECLARE nn_Cor VARCHAR(8);

    SET nn_TProfissional = IFNULL(TProfissional,-1);
    SET nn_PainScale = IFNULL(PainScale,-1);
    SET nn_DataAdmissao = IFNULL(DataAdmissao,'1111/11/11 00:00:00');
    SET nn_idCor = IFNULL(idCor,-1);
    SET nn_Cor = IFNULL(Cor,'Unknown');

    IF UEpisode IS NULL THEN SET Erro = 1;
    END IF;
    IF Erro = 1 THEN SELECT 'Número de Episódio de Urgência não pode ser nulo.';
    END IF;

START TRANSACTION;
INSERT     INTO     Fact_Triage     (Urg_Episode,     Prof_Triagem,     Pain_Scale,
FK_Date_Admission,FK_Color)     VALUES  (UEpisode,  nn_TProfissional,  nn_PainScale,
getFKdata(nn_DataAdmissao), getFKcor(nn_idCor,nn_Cor));

END $$
DELIMITER ;
```

Posteriormente, foi criado um *script* SQL, com o intuito de transitar os novos dados entre a área de retenção e o *Data Warehouse*. De forma a serem inseridos os dados com os valores corretos consoante o que já existe no *Data Warehouse*, é necessário, em alguns casos, realizar consultas ao mesmo, de forma a saber a chave primária de uma relação à qual corresponde um valor que se pretende inserir (como acontece com as relações que têm atributos auto incrementais como chaves primárias). De modo a obter estes valores de forma correta foram definidas funções auxiliares para efetuar as tais consultas ao *Data Warehouse*.

Na Figura 11, apresenta-se um exemplo de um procedimento que permite a inserção dos novos elementos na relação *Fact_Diagnosis* do *Data Warehouse*. Neste processo, é necessário recorrer a funções auxiliares, de forma a obter as chaves primárias dos elementos já inseridos no *Data Warehouse* para serem colocadas como chaves estrangeiras na tabela pretendida.

Figura 11 | Exemplo de Inserção no Data Warehouse

```
DROP PROCEDURE IF EXISTS InserirFact_Diagnosis;
DELIMITER $$
CREATE PROCEDURE InserirFact_Diagnosis()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE u INT; DECLARE d DATETIME; DECLARE d2 DATETIME;
    DECLARE dest VARCHAR(43); DECLARE reason VARCHAR(33); DECLARE codInfo VARCHAR(45); DECLARE
descInfo VARCHAR(250);
    DECLARE c VARCHAR(8);
    SET size = (SELECT COUNT(*) FROM Fact_Triage);
    SET i = 0;
    DROP VIEW  IF EXISTS vDiagnosis;
    CREATE VIEW vDiagnosis AS SELECT Urg_Episode FROM `urgency-t2`.Fact_Diagnosis;
WHILE(i<size) DO
  SET u = (SELECT Urg_Episode FROM Fact_Diagnosis LIMIT i,1);
```

```
  SET d = (SELECT Date FROM Dim_Date dD JOIN Fact_Diagnosis fD ON
dD.idDate=FD.FK_Date_Diagnosis WHERE fD.Urg_Episode=u);
  SET d2 = (SELECT Date FROM Dim_Date dD JOIN Fact_Diagnosis fD ON
dD.idDate=FD.FK_Date_Discharge WHERE fD.Urg_Episode=u);
  SET dest = (SELECT Description FROM Dim_Destination dD JOIN Fact_Diagnosis fD ON
dD.idDestination=FD.FK_Destination WHERE fD.Urg_Episode=u);
  SET reason = (SELECT Description FROM Dim_Reason dR JOIN Fact_Diagnosis fD ON
dR.idReason=FD.FK_Reason WHERE fD.Urg_Episode=u);
  SET codinfo = (SELECT Cod_Diagnosis FROM Dim_Info dI JOIN Fact_Diagnosis fD ON
dI.idInfo=FD.FK_Info WHERE fD.Urg_Episode=u);
  SET descinfo = (SELECT Description FROM Dim_Info dI JOIN Fact_Diagnosis fD ON
dI.idInfo=FD.FK_Info WHERE fD.Urg_Episode=u);
  IF(u NOT IN (SELECT * FROM vDiagnosis)) THEN INSERT INTO `urgency-t2`.Fact_Diagnosis VALUES
(u,(SELECT Prof_Diagnosis FROM Fact_Diagnosis LIMIT i,1),(SELECT Prof_Discharge FROM
Fact_Diagnosis LIMIT i,1), getFKdata2(d), getFKdestination2(dest), getFKdata2(d2),
getFKreason2(reason),getFKinfo2(codinfo,descinfo)); END IF;
  SET i = i+1;
  END WHILE;
  DROP VIEW  vDiagnosis;
END $$
DELIMITER ;
```

Por forma a tornar o processo mais intuitivo, foi criado um *script* SQL onde se encontram englobadas todas as chamadas aos procedimentos criados, bem como a limpeza da área de retenção a cada atualização.

6.2 Atualização total dos dados

A atualização de dados total, por sua vez, carateriza-se pelo total carregamento dos dados aquando da necessidade de atualização no *Data Warehouse*.

Tendo em vista a resolução deste tipo de atualização, foi aplicado um método que permite, a cada atualização, a escrita num ficheiro do formato *csv*. Numa fase posterior, seria apenas necessário a concatenação dos *csv* criados aos ficheiros do *dataset* fornecidos, antes de realizar o novo carregamento total dos dados.

Na Figura 12, apresenta-se um exemplo deste processo.

Figura 12 | Escrita no ficheiro exams.csv

```
SELECT DISTINCT a1.Urg_Episode, a2.Num_Exame, a2.Description FROM Fact_Urgency_Episodes a1
INNER JOIN Dim_Urgency_Exams a2 ON a1.FK_Urgency_Exams = a2.idUrgency_Exams
INTO OUTFILE 'exams.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';
```

7

INDICADORES CLÍNICOS

Os indicadores de avaliação do desempenho e de resultados permitem que a qualidade dos cuidados e serviços seja medida. Os indicadores de qualidade descrevem o desempenho esperado para o tratamento de determinado tipo de doente ou relacionado com os resultados de saúde [7].

Um indicador ideal é aquele que:

- É baseado em definições previamente estabelecidas, descrito exaustivamente e exclusivamente;
- Identifica poucos falsos positivos e falsos negativos (específico e sensível);
- É válido e confiável;
- Driscrimina bem os dados;
- Permite comparações úteis;
- É baseado na evidência.

Com base nas informações recolhidas, e após discussão com vários profissionais de saúde, são propostos os seguintes indicadores:

Tabela 4 | Indicadores clínicos

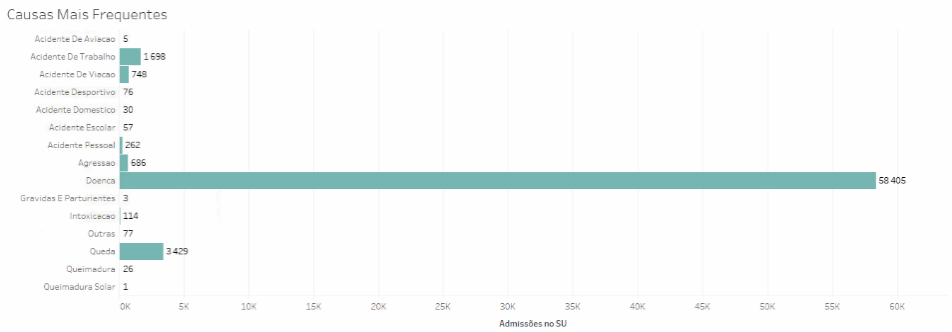
Causas mais frequentes	Patologias por distrito
Causas mais frequentes por distrito de residência	Frequência de cada intervenção clínica
Causas mais frequentes por sexo	Frequência de realização de exames
Admissões no SU por sexo	Exames realizados por grupo de patologias
Admissões no SU por faixa etária	Tempo médio entre admissão no SU e triagem
Admissões no SU por hora do dia	Tempo médio entre admissão no SU e abandono
Admissões no SU por dia da semana	Tempo de permanência no SU associado à prioridade de urgência
Destinos de alta	Tempo de permanência no SU por faixa etária
Destino de alta consoante a prioridade de urgência	Tempo de permanência por patologia
Destino dos doentes admitidos no SU	Período entre admissão na triagem e prescrição de fármacos
Destino dos doentes por faixa etária	Família de fármacos prescritos consoante o grupo de patologias
Destino dos doentes por prioridade de urgência	Atendimentos por mês
Distribuição dos doentes por prioridade	Variância de percentagem de altas
Prioridade de urgência associada à escala de dor	Afluência de prioridade verde por mês
Altas por Hora	

7.1 Relatórios Casuísticos

7.1.1 Causas mais frequentes

Este indicador permite conhecer as principais causas que levam os doentes ao SU. Observando a Figura 13, verifica-se que doença (89.01%) é a principal causa de admissão nos Serviços de Urgência, seguindo-se de quedas (5.23%) e acidentes de trabalho (2.59%).

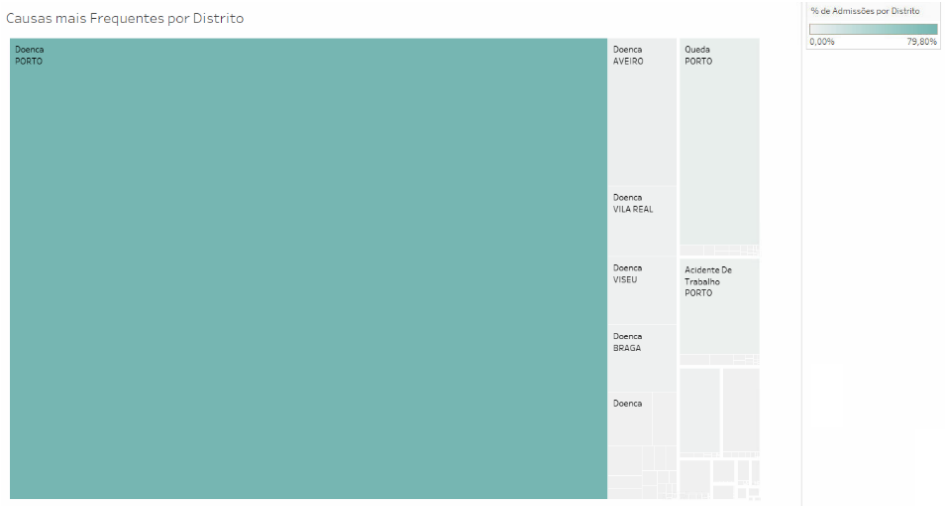
Figura 13 | Causas Mais Frequentes



7.1.2 Causas mais frequentes por distritos de residência

O hospital em estudo apresenta como áreas de influência a área metropolitana do Porto (principal), bem como os distritos de Aveiro (2.95%), Vila Real (1.41%), Viseu (1.36%) e Braga (1.35%), sendo a causa mais frequente a doença.

Figura 14 | Causas Mais Frequentes por Distritos de Residência



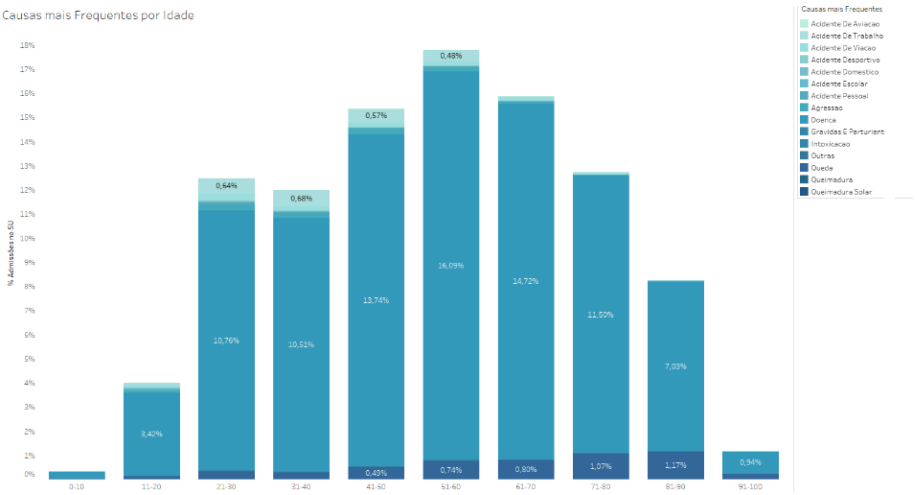
7.1.3 Causas mais frequentes por faixa etária

De forma a facilitar a análise gráfica dos dados presentes no *dataset*, foram criados grupos de idades em intervalos de dez anos.

Assim, a análise da Figura 15 permite inferir que:

- As quedas aumentam linearmente com a idade;
- Acidentes de trabalho são mais comuns nas faixas etárias em idade laboral;
- Agressões são mais frequentes nos adultos do que nos idosos (> 61 anos) ou crianças.

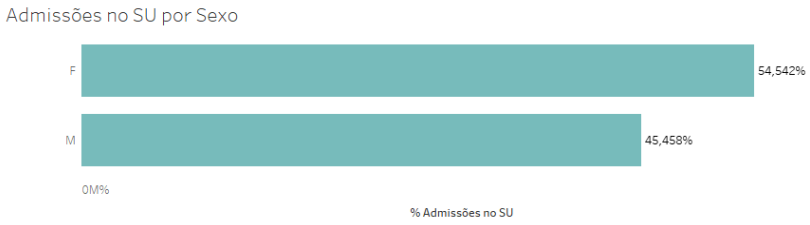
Figura 15 | Causas Mais Frequentes por Faixa Etária



7.1.4 Número de admissões no SU por sexo

Uma análise sobre a relação entre o sexo dos doentes e a sua admissão no SU revela um maior número de admissões associadas ao sexo feminino, com 54.542% das admissões.

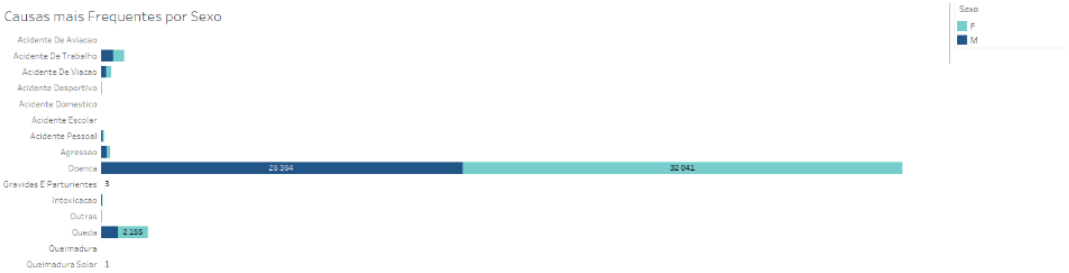
Figura 16 | Admissões no SU por sexo



7.1.5 Causas mais frequentes por sexo

O grupo mais representado foi "sexo feminino e admissão por doença" (48.83%). Verificaram-se mais quedas no sexo feminino (3.28%) e mais acidentes de trabalho (1.42%) e agressão (0.68%) no sexo masculino.

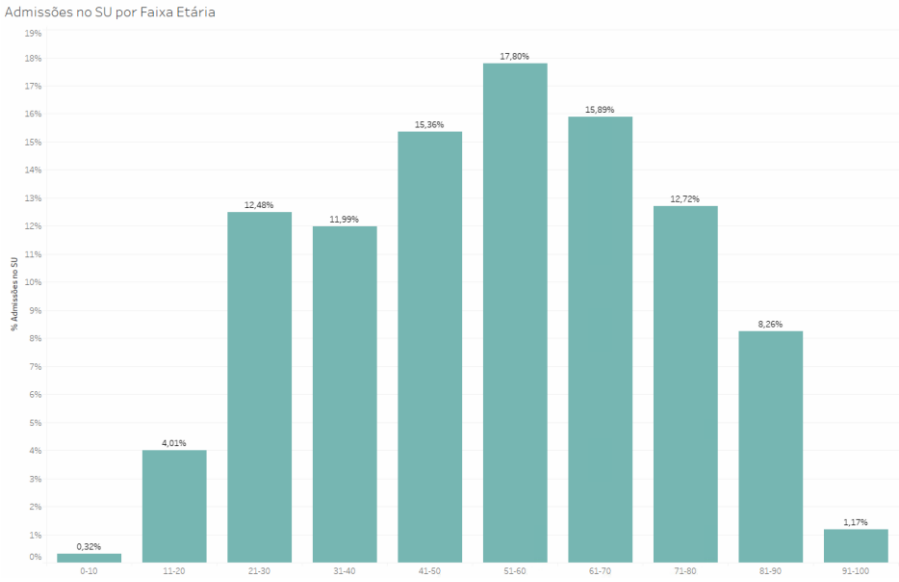
Figura 17 | Causas Mais Frequentes por Sexo



7.1.6 Admissões no SU por faixa etária

Na Figura 18 verifica-se uma distribuição normal das admissões no SU por idade. A faixa etária com mais admissões no SU encontra-se no intervalo [51-60] anos (17.80%), seguido dos intervalos [61-70], com 15.89%, e [41-50], com 15.36%. As crianças até aos dez anos são a faixa etária com menor número de admissões aos Serviços de Urgência (0.32%). Esta distribuição obedece à distribuição da faixa etária na população portuguesa.

Figura 18 | Admissões no SU por Faixa Etária



7.1.7 Admissões no SU por hora do dia

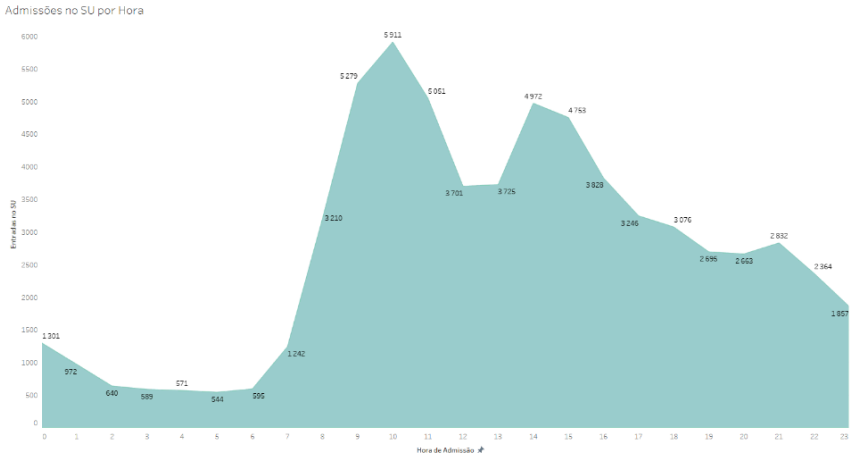
O estudo do número de admissões no SU por hora do dia permite inferir o número de profissionais de saúde necessário para cada turno – por exemplo, reforçando os turnos da manhã e da tarde em relação ao turno da noite. Do mesmo modo, poderá servir para avaliar o custo/benefício de ter determinadas especialidades em presença física/à chamada no período noturno. O mesmo se pode inferir para a disponibilidade de determinados MCDTs que requeiram operador (por exemplo, TAC, ecografia).

Pela Figura 19, verifica-se maior número de admissões durante o dia do que durante a noite. Esta pode dever-se à existência de doenças que surgem mais de manhã (como o enfarte

do miocárdio e a cólica renal) e depois de almoço (como o enfarte agudo do miocárdio e a isquemia mesentérica). Para além disso, as pessoas apercebem-se mais dos sintomas quando estão acordadas.

O período de maior admissões ocorre no intervalo entre as 7h e as 23h, ocorrendo o pico máximo às 10h (9.0%) e às 14h (7.6%). O período da noite (entre as 23h e as 6h da manhã) representa 10.8% das admissões.

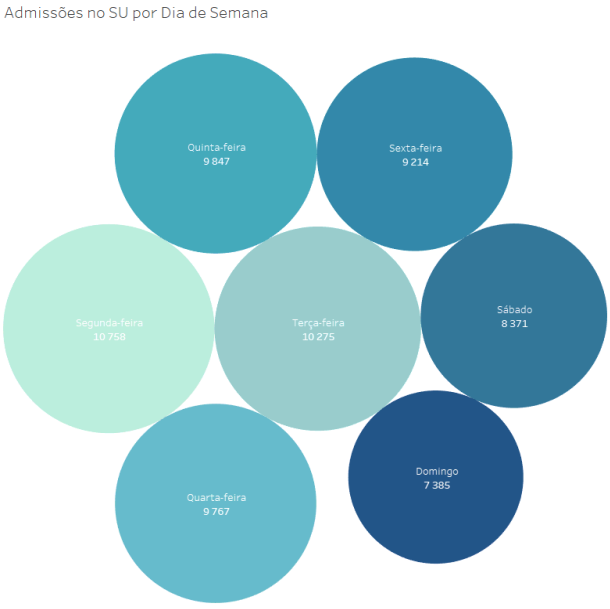
Figura 19 | Admissões no SU por hora do dia



7.1.8 Admissões no SU por dia da semana

Verifica-se (Figura 20) um maior número de admissões no início da semana (segunda-feira (16.4%) e terça-feira (15.7%)), diminuindo gradualmente ao longo da semana, sendo o número mínimo de admissões ao domingo (11.3%).

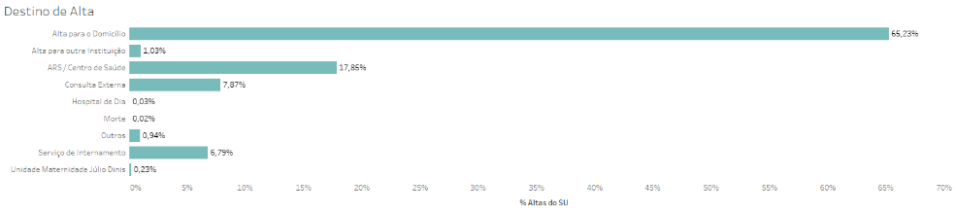
Figura 20 | Admissões no SU por Dia de Semana



7.1.9 Destinos de Alta

A maioria dos doentes tem alta para domicílio não referenciados (65.23%) e 17.85% são referenciados para ARS/Centros de Saúde. Apenas 6.79% das altas são admitidas em internamento hospitalar e 1.03% transferidas para outra instituição. De relevar a importância da colaboração entre o hospital e os cuidados de saúde primária na reavaliação destes doentes.

Figura 21 | Destinos de Alta do SU



7.1.10 Distribuição dos doentes por prioridade

O SU é diariamente confrontado com uma grande afluência de pessoas com as mais diversas sintomatologias, motivando o congestionamento e consequente sobrecarga de trabalho, prejudicando a qualidade e o tempo de resposta ao atendimento das pessoas. Por este motivo, tornou-se imprescindível a criação de um sistema de triagem inicial que promovesse o atendimento médico em função de um critério clínico, ao invés da ordem de chegada ao SU [8].

O Sistema de Triagem de Manchester (STM) tem como objetivo priorizar os doentes com maior gravidade, melhorando a qualidade da assistência prestada e aumentando a segurança dos pacientes [9].

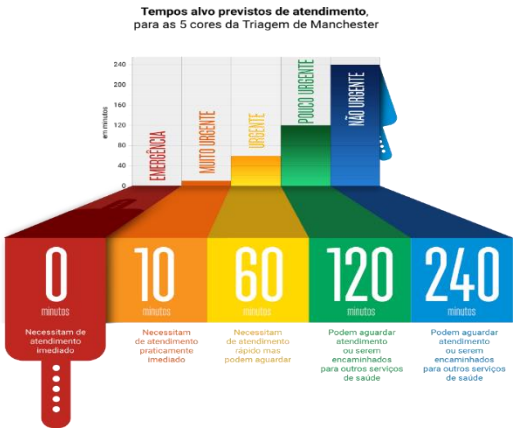
O STM tem cinco prioridades de urgência, correspondendo a cada uma a respetiva cor:

- Emergente (Vermelha);
- Muito Urgente (Laranja);
- Urgente (Amarela);
- Pouco Urgente (Verde);
- Não Urgente (Azul).

Cada uma representa um grau de gravidade e respetivo tempo de espera recomendado para a pessoa ser submetida à primeira observação médica.

A figura – apresenta os tempos alvos previstos de atendimento para as cinco cores da Triagem de Manchester, pelo Grupo Português de Triagem [10].

Figura 22 | Tempos Alvos Previstos de Atendimento



Para o conjunto de dados em estudo, são apenas consideradas as cores amarela (77.26% das admissões no SU), verde (19.30%) e laranja (3.44%).

Figura 23 | Episódios por Prioridade de Urgência

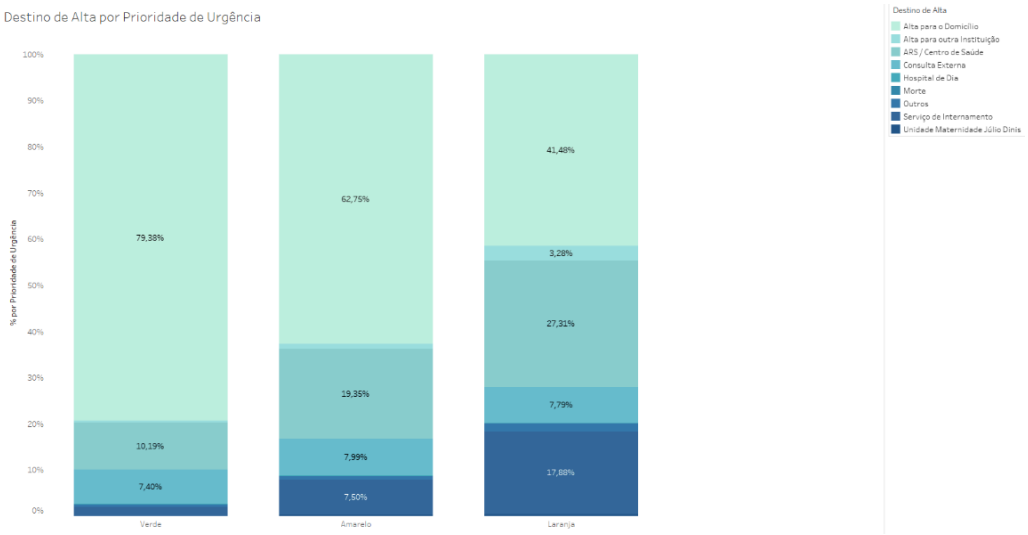
7.1.11 Destino de alta consoante a prioridade de urgência

Os doentes com prioridade verde são maioritariamente dispensados com alta para o domicílio (79.38%); 10.19% são referenciados para o Centro de Saúde e 7.40% para Consulta Externa.

Para doentes com prioridade amarela, 62.75% têm alta para domicílio, 19.35% são referenciados para consulta externa, 7.99% para Hospital de dia e 7.50% para o Internamento.

No que diz respeito aos doentes com prioridade laranja, verifica-se um aumento considerável de doentes internados (17.88), devido ao acréscimo da sua complexidade/gravidade. A taxa de mortalidade, apesar de transversalmente baixa, é maior nos doentes com pulseira laranja, demonstrando a importância do atendimento prioritário destes doentes.

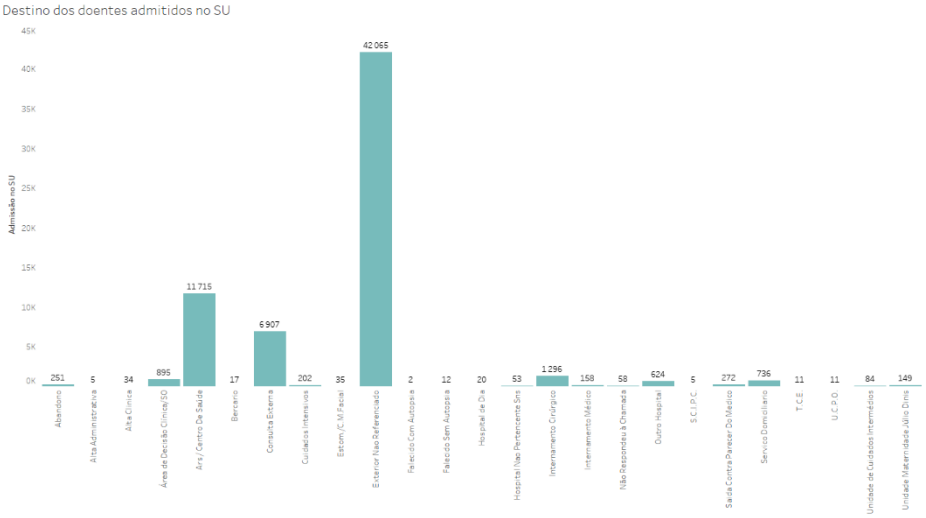
Figura 24 | Destino de Alta por Prioridade de Urgência



7.1.12 Destino dos doentes admitidos no SU

Com a análise da Figura 25, verifica-se que a grande maioria dos doentes tem alta para o exterior não referenciados (64.11%), e 17.85% são referenciados para o seu Médico de Família (ARS/Centro de Saúde).

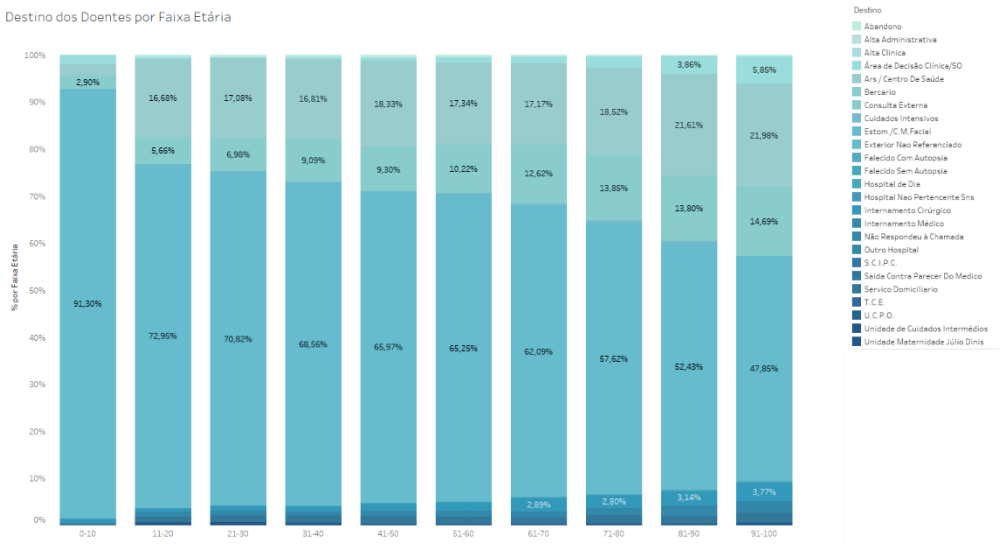
Figura 25 | Destino dos Doentes Admitidos no SU



7.1.13 Destino dos doentes por faixa etária

A maioria dos doentes tem alta para o exterior, não sendo referenciados para nenhum serviço de saúde (i.e, Centro de Saúde, Consulta hospitalar, etc), sendo este achado transversal a todas as idades.

Figura 26 | Destino dos Doentes por Faixa Etária



7.1.14 Destino dos Doentes por Prioridade de Urgência

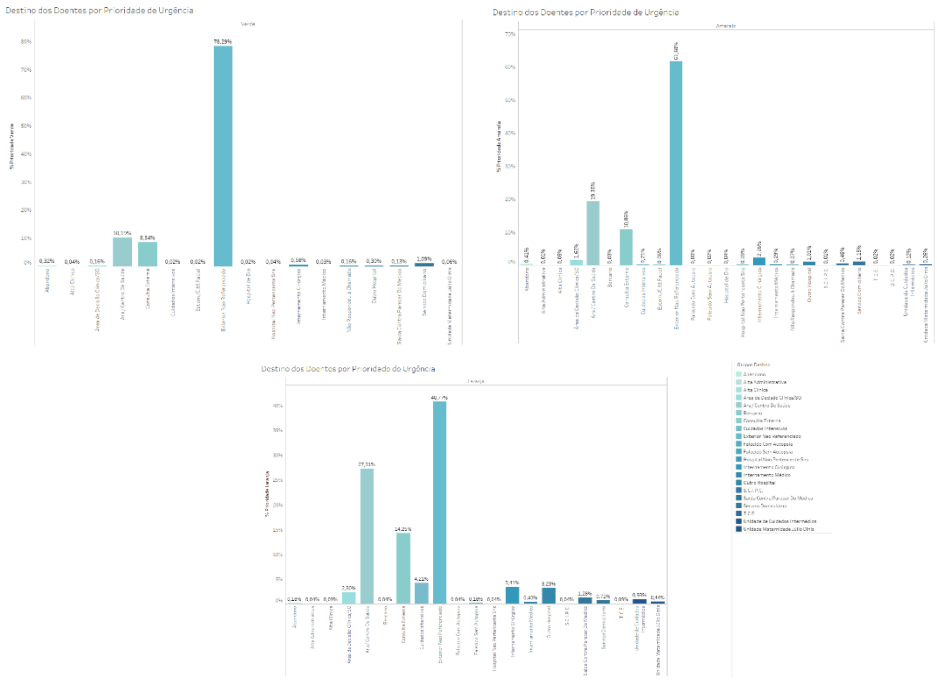
Para uma melhor análise deste indicador, foram agrupados os diversos destinos dos doentes em grupos mais abrangentes. Na Tabela 5, estão discriminados os grupos criados.

Tabela 5 | Grupos de Destinos

Destinos	Destinos - Detalhado
Consulta Externa	Cardiologia, CE, Dermatologia, Neurologia, entre outros
Internamento Cirúrgico	Cirurgia (1,2), Cirurgia Vascular, ORL
Cuidados Intensivos	Cuidados Intensivos, U.C.I.C, U.C.I.P
Hospital de Dia	HD
Internamento Médico	Medicina (1,2)
Unidade de Cuidados Intermédios	Unidade Cuidados intermédios de Urgência, Unidade Intermédia Médica

Verifica-se uma maior referenciação para Centro de Saúde/Consulta à medida que a prioridade aumenta; O internamento é maior nas admissões de maior prioridade. Para além disso, verifica-se uma necessidade crescente de recurso à área de decisão clínica linearmente com o aumento da prioridade. Não se verificaram óbitos na prioridade verde.

Figura 27 | Destino dos doentes por Prioridade de Urgência



7.1.15 Prioridade de Urgência associada à escala de dor

A dor é um sintoma que acompanha, de forma transversal, a generalidade das situações patológicas que requerem cuidados de saúde. A avaliação e registo da intensidade da dor, pelos profissionais de saúde, tem de ser feita de forma contínua e regular, de modo a otimizar a terapêutica e melhorar a qualidade de vida do doente.

A Escala Numérica (EN) é uma escala de mensuração da intensidade de dor validada internacionalmente. Esta escala consiste numa régua dividida em onze partes iguais, numeradas sucessivamente de 0 a 10. Pretende-se que o doente avalie a intensidade da sua dor numa classificação numérica, sendo que a 0 corresponde a classificação "Sem Dor" e a 10 a classificação "Dor máxima" [11].

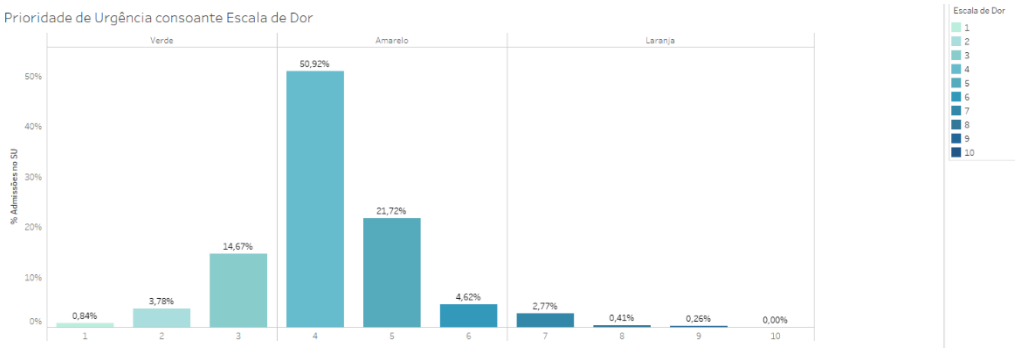
Figura 28 | Escala Numérica de Mensuração da Intensidade de Dor

0	1	2	3	4	5	6	7	8	9	10
Nenhuma		Pouca			Razoável			Média		Excessiva

Analisando a Figura 29, a atribuição da prioridade às pessoas admitidas no SU vai de encontro com a intensidade de dor indicada pelo mesmo. Aproximadamente 92% das pessoas apresentava dor ≤ 5, sendo que nenhum doente com dor ≥ 7 foi triado com cor verde ou amarela.

Um indicador importante seria a comparação da intensidade de dor com o tempo de espera no SU, não existindo, no entanto, dados que permitam esta avaliação.

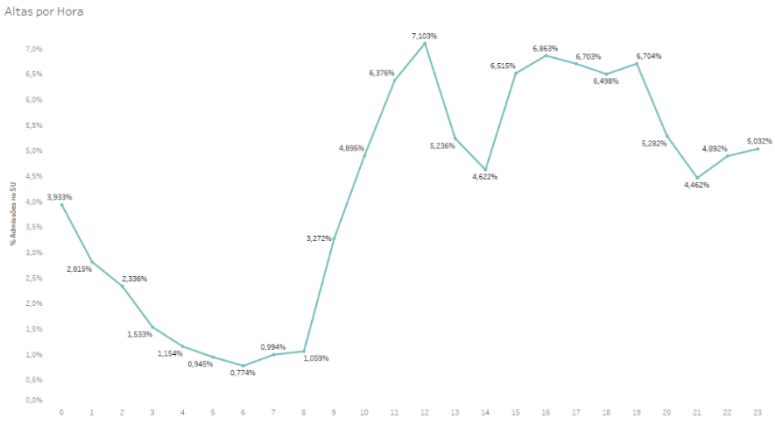
Figura 29 | Prioridade de Urgência consoante Escala de Dor



7.1.16 Altas por Hora

Verifica-se um atraso entre o pico de admissões e de altas por hora (com realce nas 14h onde as admissões por hora alcançam um pico máximo e as altas por hora um pico mínimo). Seria necessário avaliar uma terceira variável, o tempo de espera até atendimento. Na eventualidade de este ser elevado nestas discrepâncias, uma possível solução estaria na colocação de mais profissionais nas horas de maior afluência no SU.

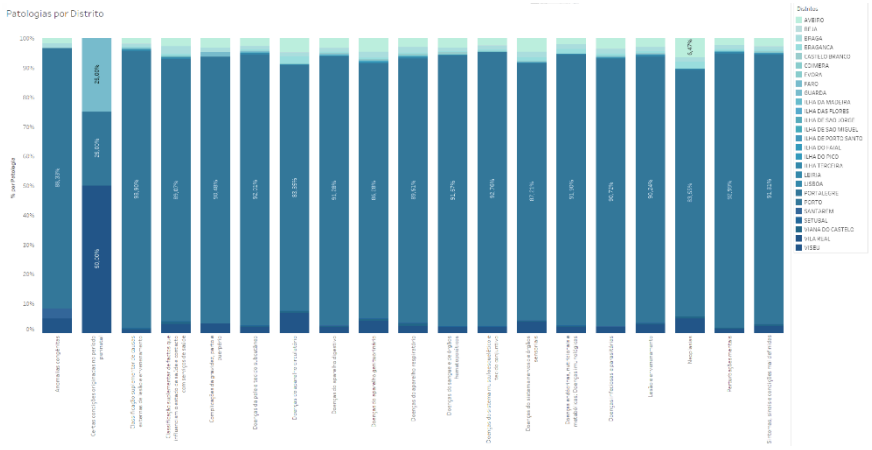
Figura 30 | Altas por Hora



7.1.17 Patologias por Distrito

Em determinadas patologias (“Certas condições originadas no período perinatal”), há uma maior variedade de distritos implicados, o que demonstra que este hospital é hospital de referência para determinadas especialidades médicas.

Figura 31 | Patologias por Distrito

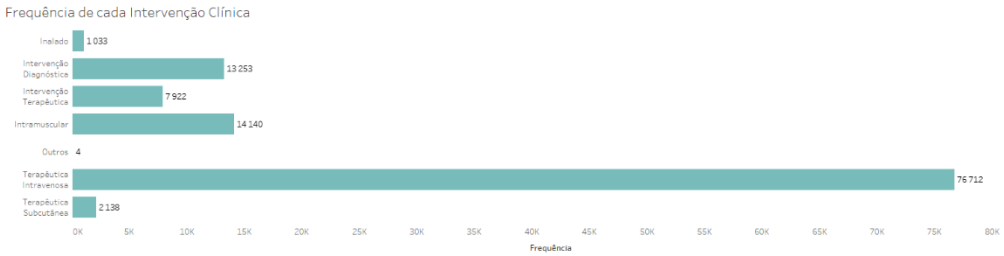


7.2 Relatórios de Qualidade de Dados

7.2.1 Frequência de cada intervenção clínica

As intervenções clínicas mais frequentes são a terapêutica intravenosa (66.59%), seguido de intervenções intramusculares, com 12.27%.

Figura 32 | Frequência de cada Intervenção Clínica



7.2.2 Frequência de realização de exames

Dada a grande diversidade de tipos de exames no conjunto de dados, foi necessária a criação de grupos mais abrangentes, de forma a permitir a análise dos seus dados. Na Tabela 6, são apresentados os grupos de exames criados, bem como alguns dos exames que estes englobam.

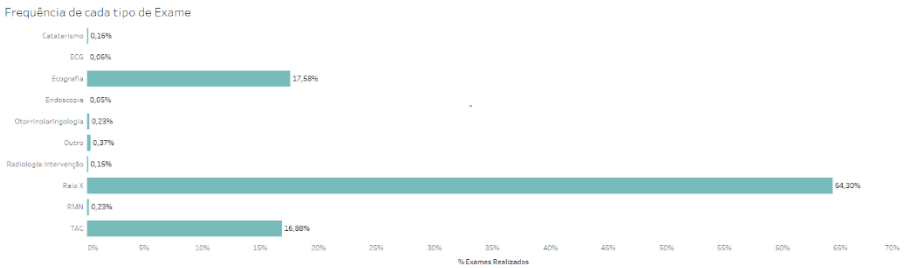
Tabela 6 | Grupo de Exames Realizados

Exame	Exame – Detalhado
Cateterismo	Coronariografia, Angioplastia, Cateterismo
ECG	ECG, Holter, Monitorização electrocardiográfica
Ecografia	Doppler, Exame Ultrassonográfico, Ecocardiograma
Endoscopia	Endoscopia, Cintigrafia, Colonoscopia, Polipectomia
Otorrinolaringologia	Audiometria, Timpanograma, ECA, Reflexos acústicos
Outro	Biomicroscopia, Tomografia, Cistometria, Biometria
Radiologia Intervenção	Drenagem, Trombectomia, Embolização arterial
Raio X	Radiologia, Sios perinasais, Tomossíntese, Bacia

RMN	RM, Suplemento para Angiografia RM com gadolínio
TAC	Angio TC, Tac, TC

ECG – Electrocardiograma; TAC – Tomografia Axial Computarizada; ECA – Estudo de Condutividade Auditiva; RMN – Ressonância Magnética Nuclear; RM – Ressonância Magnética.

Figura 33 | Frequência de Realização de Cada tipo de Exame



O tipo de exames mais requisitado é o Raio-X, correspondendo a 64.30% dos exames realizados, seguido das ecografias (17.58%) e TAC (Tomografia Axial Computarizada) (16.88%).

É necessário avaliar a relação risco-benefício entre os diferentes tipos de exames realizados. As ecografias são mais inócuas para o doente, uma vez que não envolvem radiação e são mais baratas que a realização de um TAC. No entanto, a sua utilização implica a presença de um radiologista no local. Pode existir uma potencial subutilização de ecografia por falta de radiologistas. Por este motivo, a implementação de aparelhos de TAC permite descentralizar as técnicas diagnósticas, por possibilidade de enviar os exames para centros especializados.

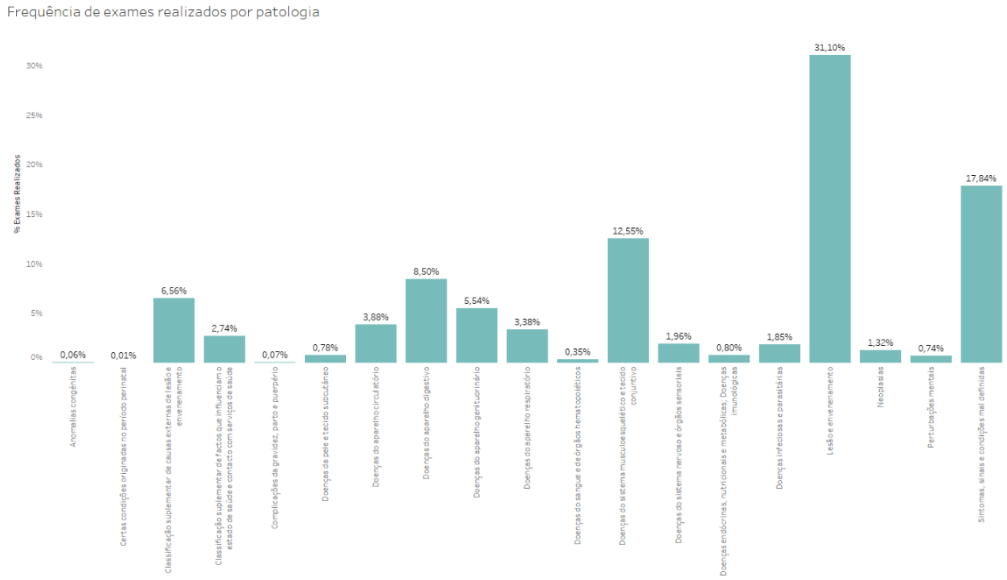
Para exames mais superespecializados, como radiologia de intervenção e cateterismo, é crucial avaliar a necessidade de disponibilidade em todos os hospitais ou de centralizar. É preciso ter em consideração não só os custos como os tempos para tratamento de certas patologias (por exemplo, no enfarte agudo do miocárdio, deverá haver uma sala de hemodinâmica a menos de 120 minutos para poder realizar cateterismo emergente).

7.2.3 Exames Realizados por Grupo de Patologias

O grupo nosológico que requisita mais exames é “Lesão e Envenenamento” (31.01%), seguido de “Sintomas, Sinais e Condições Mal Definidas” (17.84%) e “Doenças do Sistema Musculoesquelético do Tecido Conjuntivo”, com 12.55%.

Os exames disponíveis devem ter em conta estas patologias. Para além disso, é necessário garantir a disponibilidade de profissionais que saibam interpretar os exames (por exemplo, um pedido de Raio-X durante a noite, por parte de um internista, só será interpretado por um médico ortopedista na manhã seguinte levando ao aumento do tempo de permanência do doente no SU).

Figura 34 | Exames Realizados por Grupo de Patologias



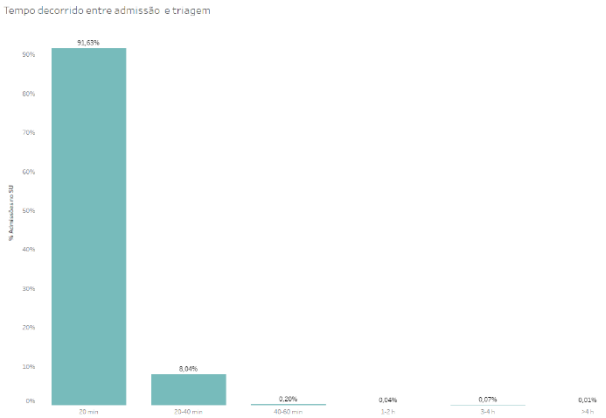
7.2.4 Tempo médio entre admissão no SU e triagem

Idealmente, uma pessoa deve ser triada imediatamente após a sua admissão no Serviço de Urgência.

Verifica-se, no entanto, que, apesar de 99% das pessoas admitidas no SU serem triadas na primeira hora após admissão, 0.07% das pessoas triadas com pulseira amarela (0.089% com pulseira verde e 0.095% com pulseira laranja) são admitidas na triagem apenas após três horas da sua admissão no SU.

É importante tentar perceber a causa dos períodos de espera superiores a 24 horas, de forma a encontrar soluções para reduzir o tempo de espera (um doente triado laranja pode ter um tempo de espera de 40 minutos quando se pretende um atendimento de 10 minutos).

Figura 35 | Tempo Médio entre Admissão no SU e Triagem

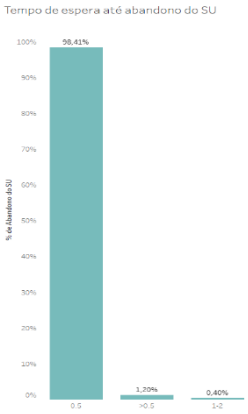


7.2.5 Tempo médio entre admissão no SU e abandono

Foram registados 251 casos de abandono do SU. Destes, 98.41% ocorrem na primeira meia hora de espera, 1.20% até concluir a primeira hora e 0.4% entre a primeira e segunda horas.

É necessário estudar as possíveis causas de abandono para conseguir reduzir a sua prevalência.

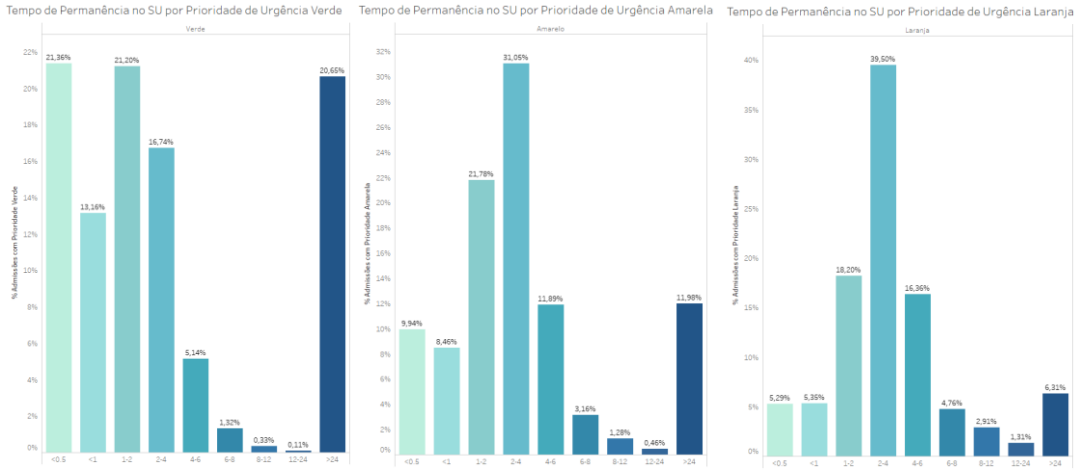
Figura 36 | Tempo Médio entre Admissão no SU e Abandono (%)



7.2.6 Tempo de permanência no SU associado à prioridade de urgência

Verifica-se que 20.65% dos doentes com pulseira verde permanecem mais de 24h no SU. Tratando-se de doentes com uma reduzida complexidade, é crucial analisar as causas que levam a este elevado nível de permanência no SU.

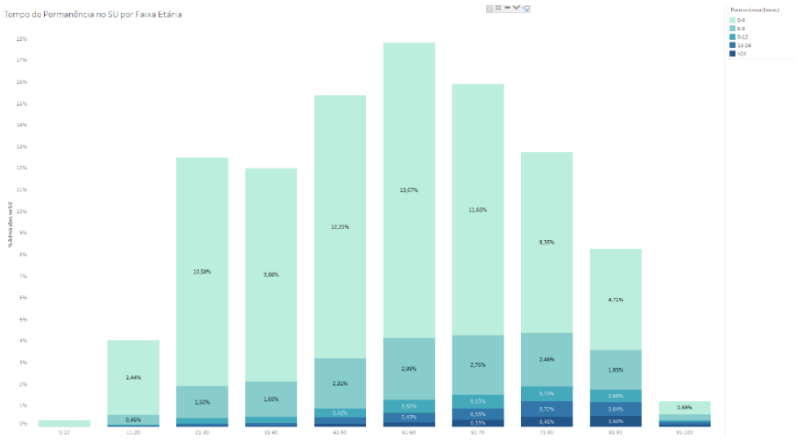
Figura 37 | Tempo de Permanência no SU associado à Prioridade de Urgência (em horas)



7.2.7 Tempo de permanência no SU por faixa etária

Observa-se uma maior proporção de doentes que permanece no SU mais de 4 h no grupo dos 80 e 90 anos. Por este motivo, deve considerar-se aumentar a linha de comunicação entre familiares e profissionais de saúde, no caso de doentes idosos/frágeis/fisicamente dependentes. Para além disso, aumentar o local de observação e condições para observação de doentes permanentemente alectuados.

Figura 38 | Tempo de Permanência no SU por Faixa Etária

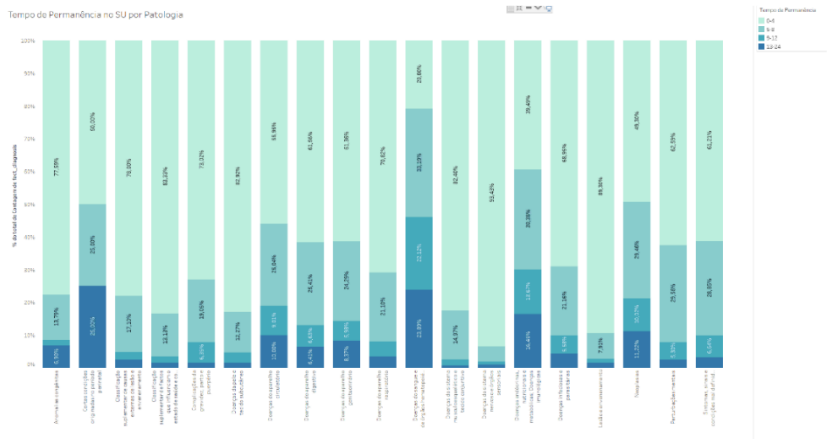


7.2.8 Tempo de permanência por patologia

As patologias que requerem mais tempo de permanência no SU são "Certas condições originadas no período perinatal", onde 25% permanecem entre 13 e 24 horas; "Doenças do sangue e de órgãos hematopoiéticos" com 23.89% e "Doenças endócrinas, nutricionais e metabólicas; Doenças imunológicas" com 16.46%, no mesmo período.

Sugere-se ter em consideração os cuidados necessários para atender doentes com estas patologias, de modo a conseguir minorar o tempo de espera e orientar com brevidade estes doentes para outro local.

Figura 39 | Tempo de Permanência por Patologia

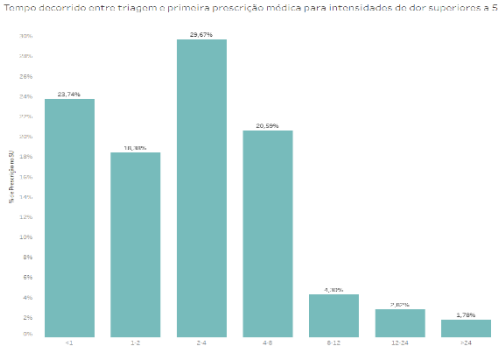


7.2.9 Período entre admissão na triagem e prescrição de fármacos

Considerando dor intensa como ≥ 5 na EN, verifica-se que uma grande percentagem de doentes esperou mais que 1 hora até ser sido administrada analgesia.

Uma potencial abordagem a este problema passa pela sensibilização dos profissionais de saúde para dor (posters, formação) e a facilidade de comunicação entre doentes e profissionais de saúde.

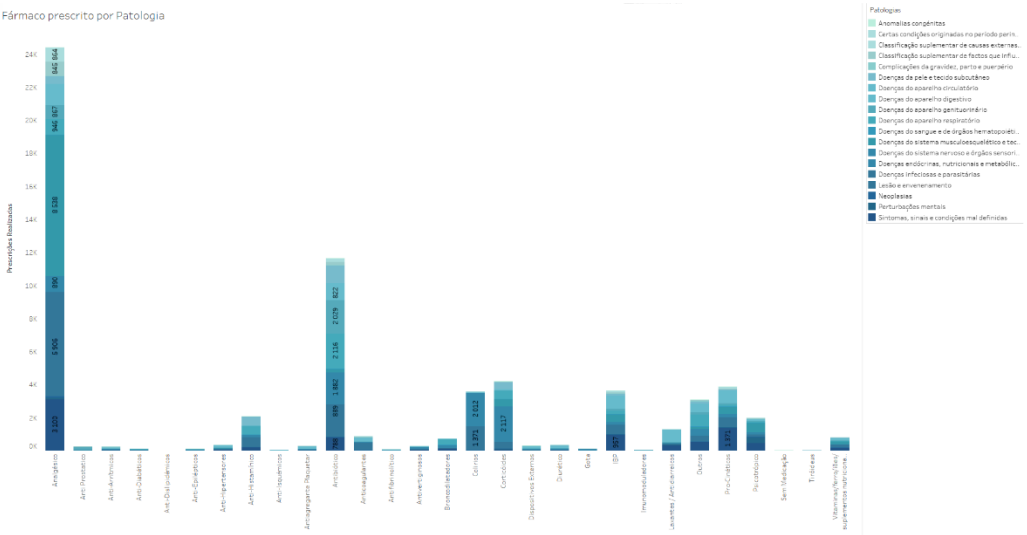
Figura 40 | Período entre Admissão na Triagem e Prescrição de Fármacos



7.2.10 Família de fármacos prescritos consoante o grupo de patologias

Os analgésicos (37.47%) são o grupo de fármacos mais prescritos no SU, seguido dos antibióticos (17.9%) e corticoides (6.48%).

Figura 41 | Família de Fármacos prescritos consoante o Grupo de Patologias



7.2.11 Atendimentos por Mês

Na Figura 42, é possível verificar picos de atendimento nos meses de março e agosto. Seria necessário compará-los com os tempos de espera para atendimento, também por mês. Caso o tempo de espera se verifique mais elevado do que os atendimentos, é crucial a contratação de mais profissionais de saúde para os meses com menos afluência, ou a colocação de tarefeiros ou aumento de horas extra. Uma vez que o conjunto de dados termina em meados de outubro, foram desconsiderados os meses de outubro, novembro e dezembro.

Figura 42 | Atendimentos por Mês

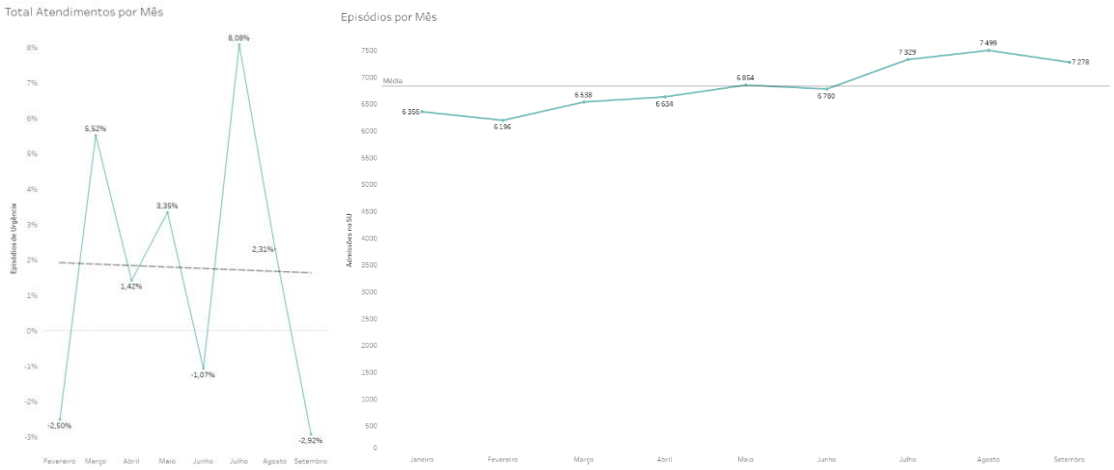


Tabela 7 | Média do número de atendimentos, por mês

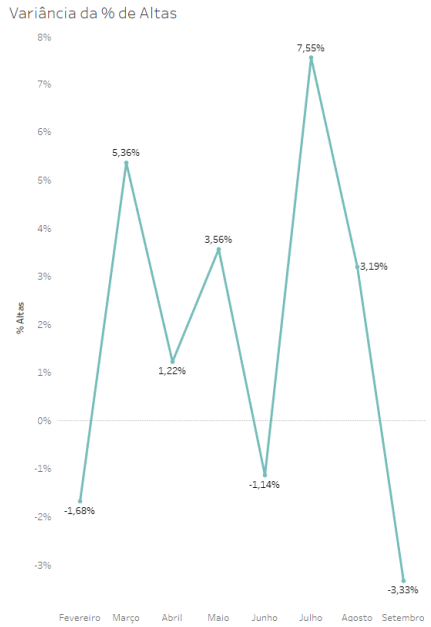
ARS Norte [12]	**Hospital em Estudo**	**Diferença**
13557	6356	-7201
12470	6196	-6274
12826	6538	-6288
12435	6634	-5801
13094	6854	-6240
12524	6780	-5744
13157	7329	-5828
13575	7499	-6076
9472	7278	-2194

O número de atendimentos por mês do hospital da amostra é muito inferior à média de atendimentos de hospitais da zona Norte. É de notar que existem várias tipologias de hospitais e, desconhecendo a tipologia deste hospital, não é possível realizar uma analogia se realmente está abaixo ou acima da média de atendimentos, dependente do tipo de hospital.

7.2.12 Variância de Percentagem de Altas

O número de altas por mês está em consonância com o número de admissões. Este indicador permite inferir a variação do fluxo de doentes ao longo do ano.

Figura 43 | Variância de Percentagem de Altas por Mês



7.2.13 Afluência de Prioridade Verde por Mês

Um indicador de extrema importância na verificação de qualidade é a afluência de doentes com prioridade verde admitidos no SU, dado que este só deveria admitir prioridades amarelas e superiores (os doentes triados com pulseira verde são considerado de menor urgência, devendo ser atendidos no Centro de Saúde).

Figura 44 | Afluência de Prioridade Verde por Mês

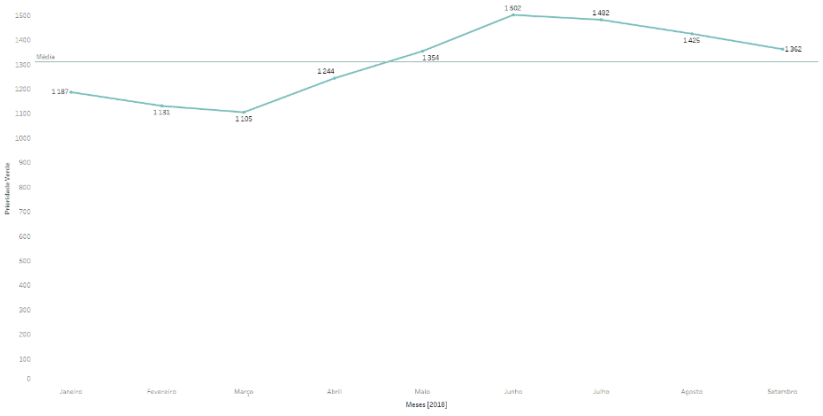


Tabela 8 | Afluência de Prioridade Verde por Mês, em percentagem

Meses	ARS Norte (%) * [12]	Hospital em Estudo (%)	Diferença (%)
Janeiro	41.0	18.7	-22.3
Fevereiro	40.8	18.3	-22.5

Março	41.3	16.9	-24.4
Abril	41.3	18.8	-22.5
Maio	41.4	19.8	-21.6
Junho	41.6	22.2	-19.4
Julho	41.7	20.4	-21.3
Agosto	41.7	19.0	-22.7
Setembro	41.7	18.7	-23.0

***Percentagem conjunta de verdes + brancos + azuis, assume-se que o número de azuis e brancos é praticamente desprezível**

Verifica-se um acréscimo de admissão de doentes com prioridade verde no segundo semestre do ano, tendo atingindo o pico em junho, com 1502 admissões. Este hospital apresenta significativamente menos admissões com prioridade verde em comparação com a média dos hospitais da zona norte, com uma diferença por volta dos 20 pontos percentuais.

7.2.14 Atendimentos urgentes com internamento

Figura 45 | Atendimentos urgentes com internamento

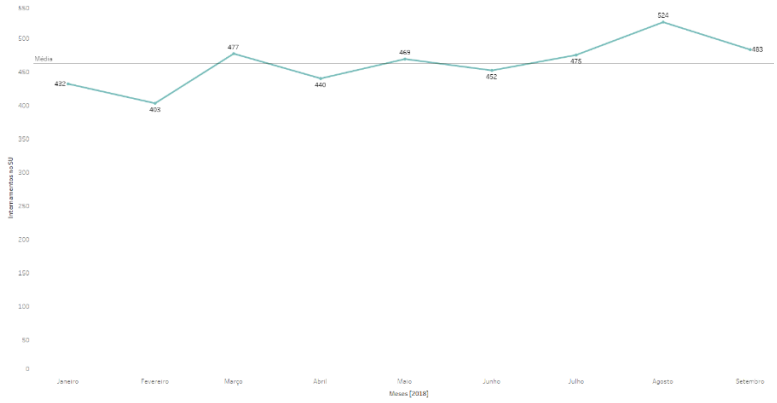


Tabela 9 | Taxa de atendimentos urgentes com internamento, por mês

Meses	Total SNS (%) [12]	Hospital em Estudo (%)	Diferença (%)
Janeiro	8.4	6.8	-1.6
Fevereiro	8.3	6.5	-1.8
Março	8.4	7.3	-1.1
Abril	8.3	6.6	-1.7
Maio	8.2	6.8	-1.4
Junho	8.2	6.7	-1.5
Julho	8.1	6.5	-1.6
Agosto	8.1	7.0	-1.1
Setembro	8.2	6.6	-1.6

A taxa de internamentos das admissões no SU varia entre 6.5% e 7.3% do total de admissões no SU.

Verifica-se que o hospital em estudo apresenta transversalmente ao longo dos meses analisados menor taxa de internamento hospitalar em relação à média dos hospitais pertencentes ao SNS (-1.1% a -1.8%).

7.3 Painéis com indicadores

Figura 46 | Painel Triagem

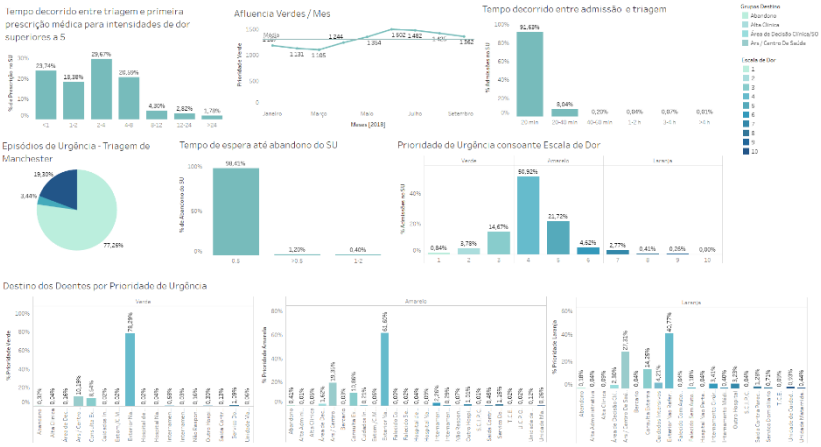


Figura 47 | Painel Diagnóstico I

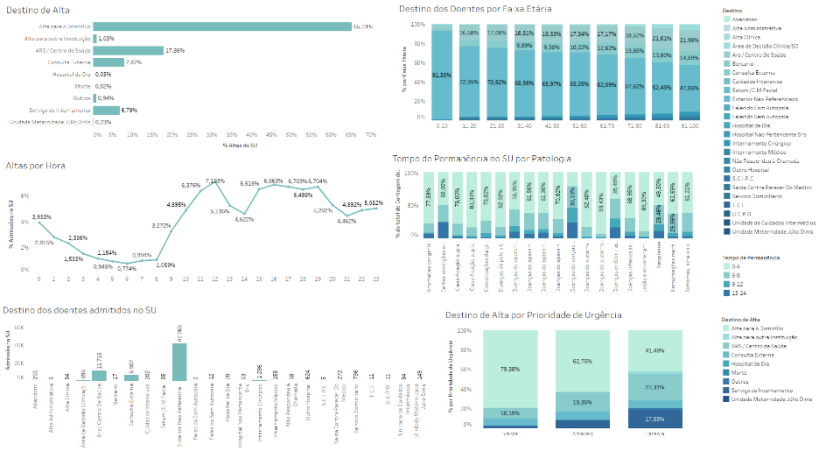


Figura 48 | Painel Diagnóstico II

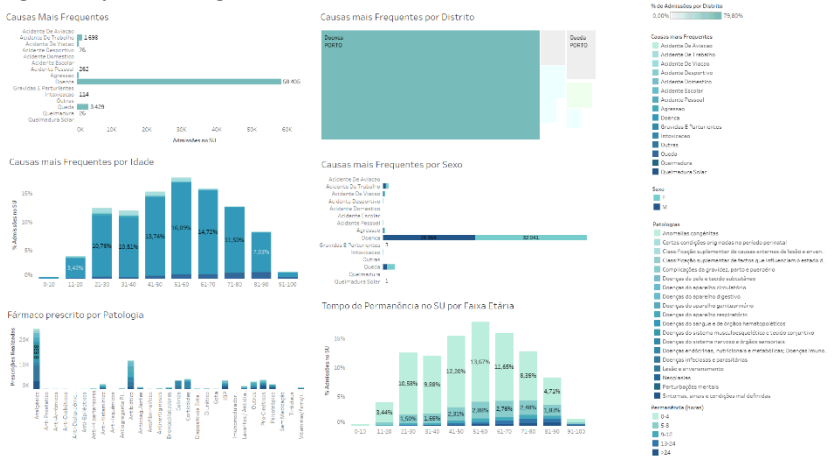


Figura 49 | Painel Episódios I

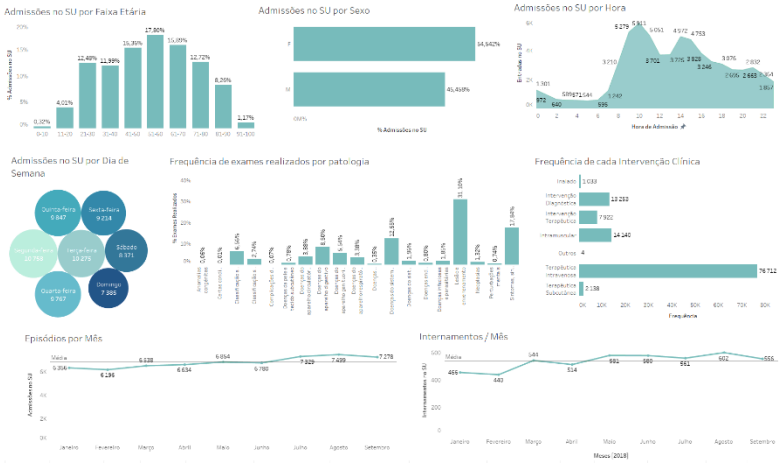


Figura 50 | Painel Episódios II

8

APLICAÇÃO PARA MONITORIZAÇÃO DO SERVIÇO DE URGÊNCIA

Foi desenvolvida uma aplicação modelo para monitorização do Serviço de Urgência. Assim, é possível analisar os diversos indicadores anteriormente apresentados, facilitando a gestão do Serviço de Urgência. Apresentam-se, em seguida, algumas páginas esqueleto desta possível aplicação *web*.

Figura 51 | Página Inicial

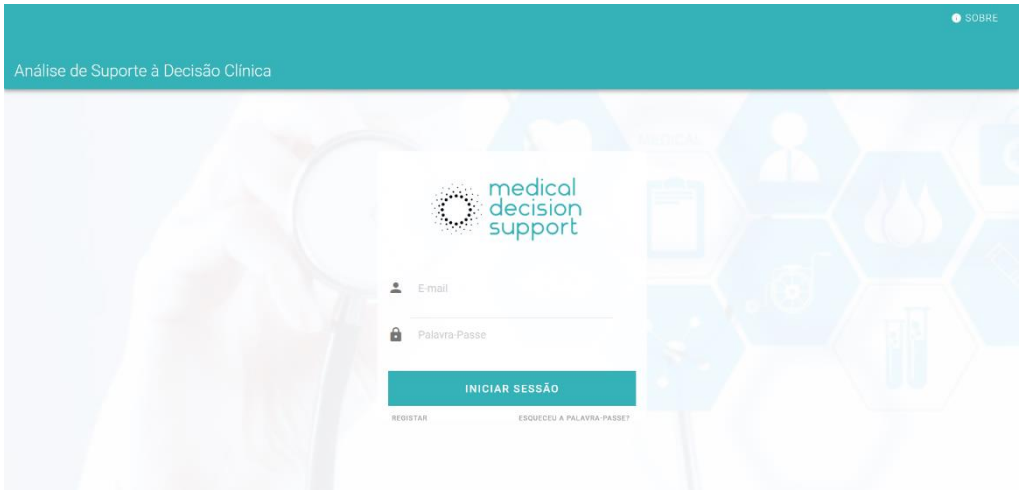


Figura 52 | Menu Principal

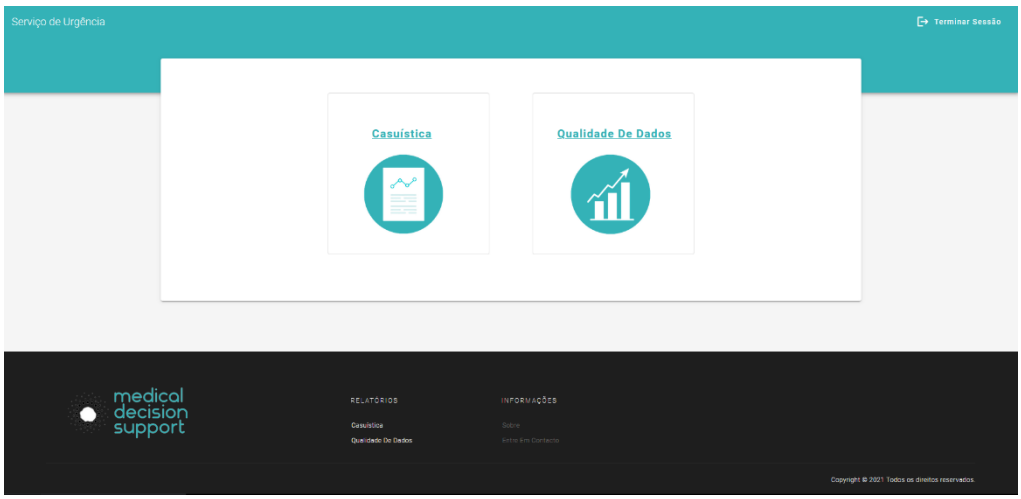


Figura 53 | Página Casuística I

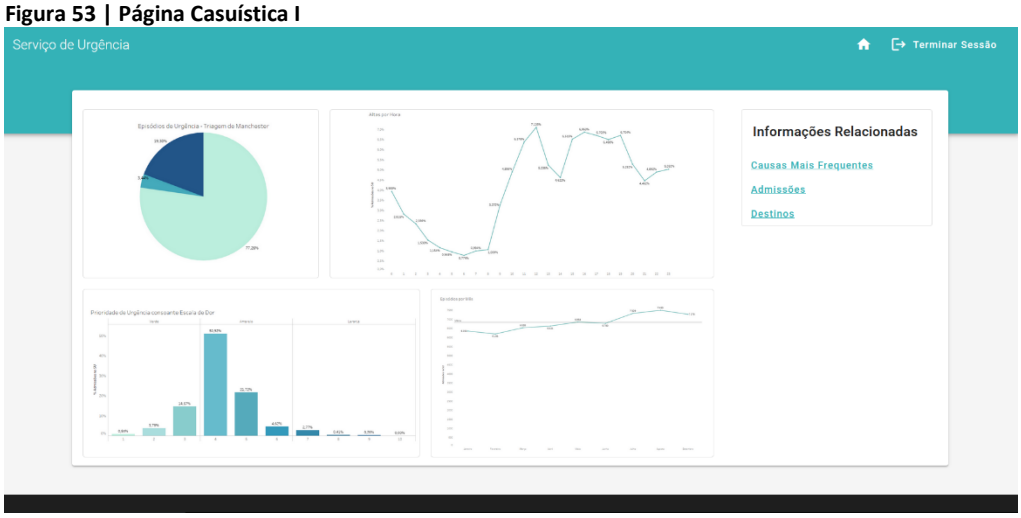


Figura 54 | Página Casuística II

9

CONCLUSÃO

Desde o momento em que o presente projeto foi atribuído, os elementos abraçaram este com entusiasmo por ser um tema de preocupação geral. O grupo tem consciência que os processos tratados e descritos durante a elaboração deste podem ter um impacto enorme no capítulo da saúde, aumentando a motivação para a sua elaboração.

A realização deste trabalho foi um grande desafio tendo em conta essencialmente a quantidade de dados fornecida. Esta dificultou principalmente o processo de importação dos dados para o *Data Warehouse*. Este problema foi, por outro lado, visto pelo grupo como uma oportunidade de conhecer melhor a ferramenta *Talend* introduzida durante as aulas práticas da Unidade Curricular. Esta ferramenta permitiu tratar do processso de importação de uma forma mais rápida do que se estava a verificar até então. Um outro contratempo esteve na dificuldade de aplicar algumas ferramentas nas máquinas dos elementos do grupo, atrasando desta forma, a evolução do projeto.

A colaboração de profissionais de saúde permitiu uma seleção e análise mais aprofundada dos indicadores mais importantes para suporte à decisão clínica de um Serviço de Urgências, apesar de desconhecido o hospital em estudo e consequentes processos de funcionamento.

Numa perspetiva futura, o grupo considera que seria interessante o aprimoramento da interface web desenvolvida, de forma a apresentar os dados consoante preferência do utilizador.

Por fim, o grupo viu este projeto como uma excelente oportunidade de aprendizagem, dado que foram utilizadas ferramentas até então desconhecidas, que se considera serem bastante utéis em diversas áreas e situações.

BIBLIOGRAFIA

[1] K. &. Ross, The Data Warehouse Toolkit, Indianapolis: Wiley Publishing, Inc, 2002.

[2] K. &. Caserta, The Data Warehouse ETL Toolkit, Indianapolis: Wiley Publishing, Inc, 2004.

[3] "What is business intelligence? Your guide to BI and why it matters," [Online]. Available: https://www.tableau.com/learn/articles/business-intelligence.

[4] "What is ETL," Talend, [Online]. Available: https://www.talend.com/resources/what-is-etl/.

[5] "Administração Central do Sistema de Saúde, IP," [Online]. Available: https://www2.acss.min-saude.pt/DepartamentoseUnidades/DepartamentoGest%C3%A3oeFinanciamentoPrestSa%C3%BAde/Codifica%C3%A7%C3%A3oCl%C3%ADnica/tabid/358/language/pt-PT/Default.aspx.

[6] "Serviço Nacional de Saúde - ICD," [Online]. Available: https://www.ctc.min-saude.pt/2018/11/08/icd-9-cm/.

[7] J. Mainz, "Defining and Classifying Clinical Indicators for Quality Improvement," *International Journal for Quality in Health Care,* vol. 15, nº 6, pp. 523-530, 2003.

[8] M. M. C. d. Carvalho, "O Sistema de Triagem de Manchester e a Avaliação da Pessoa com Dor," *Escola Superior de Enfermagem de Coimbra,* novembro 2016.

[9] e. a. Juliana Barros Becker, "Triagem no Serviço de Emergência: Associação entre as suas Categorias e os Desfechos do Paciente," *Journal of Scholl of Nursing,* 2006.

[10] "Grupo Português de Triagem," [Online]. Available: http://www.grupoportuguestriagem.pt/index.php?option=com_content&view=article&id=4&Itemid=110.

[11] D. G. d. Saúde, "A Dor como 5º Sinal Vital. Registo Sistemático da Intensidade da Dor," 2003.

[12] "Monitorização Mensal Hospitais - Serviço Nacional de Saúde," [Online]. Available: https://benchmarking-acss.min-saude.pt/MH_ProdRacioEficUrgenciaDashboard.

[13] "Serviço Nacional de Saúde," [Online]. Available: https://benchmarking-acss.min-saude.pt/MH_ProdRacioEficUrgenciaDashboard.

A

ANEXOS

Anexo 1 | Código *python* para tratamento das datas de nascimento

```
import pandas as pd
uE = pd.read_csv("urgency_Episodes.csv")
for INDEX,row in uE.iterrows():
    if ( int(uE.iloc[INDEX]['DATE_OF_BIRTH'][0:4]) >= 2018 ) :
        size = len(uE.iloc[INDEX]['DATE_OF_BIRTH'])
        newString = '19' + uE.iloc[INDEX]['DATE_OF_BIRTH'][2:size]
        uE.at[INDEX,'DATE_OF_BIRTH'] = newString
uE.to_csv("urgency_Episodes.csv", INDEX = False,sep=';')

#FIX URGENCY EXAMS, algumas descrições estavam vazias
exams = pd.read_csv("urgency_exams.csv", sep=';')
exams['DESC_EXAM'] = exams['DESC_EXAM'].fillna('Sem descrição')
exams.to_csv("urgency_exams.csv", INDEX = False, sep=";")
```

Anexo 2 | Código python para tratamento dos dados nulos

```
import pandas as pd
episodes = pd.read_csv("urgency_Episodes.csv",sep=';')
exams = pd.read_csv("urgency_exams.csv", sep=';')
procedures = pd.read_csv("urgency_procedures.csv",sep=';')
prescriptions = pd.read_csv("urgency_prescriptions.csv",sep=';')

episodesID = episodes['URG_EPISODE'].tolist()
examsID = exams['URG_EPISODE'].tolist()
proceduresID = procedures['URG_EPISODE'].tolist()
prescriptionsID = prescriptions['URG_EPISODE'].tolist()

l1 = list(SET(episodesID) - SET(examsID))
for i in l1:
  insert = pd.DataFrame({"URG_EPISODE":[i],"NUM_EXAM":["Sem Exame"], "DESC_EXAM":["Sem Exame"]})
  exams = pd.concat([exams, insert], ignore_INDEX = True)
exams = exams.iloc[:,::-1]
exams.to_csv("urgency_exams_2.csv", INDEX = False, sep = ';')

l2 = list(SET(episodesID) - SET(proceduresID))
for i in l2:
   insert = pd.DataFrame({ "URG_EPISODE":[i],"ID_PROFESSIONAL":[-1],"DT_PRESCRIPTION":["1111-11-11 00:00:00"],
                           "ID_PRESCRIPTION" : [-1], "DT_BEGIN" : ["1111-11-11 00:00:00"], "DT_CANCEL" : [0],
                           "ID_PROFESSIONAL_CANCEL" : [-1], "ID_INTERVENTION" : [-1], "DESC_INTERVENTION" :
["Sem Intervenção"]})
   procedures = pd.concat([procedures, insert], ignore_INDEX = True)
procedures = procedures.iloc[:,::-1]
procedures.to_csv("urgency_procedures_2.csv", INDEX = False, sep = ';')

l3 = list(SET(episodesID) - SET(prescriptionsID))
for i in l3:
 insert = pd.DataFrame({"URG_EPISODE":[i],"COD_PRESCRIPTION":[-1],"ID_PROF_PRESCRIPTION":["-1"],
                    "DT_PRESCRIPTION" : ["1111/11/11 00:00:00"], "COD_DRUG" : [-1], "QT" : [0],
                     "DESC_DRUG" : ["Sem Medicação"]})
    prescriptions = pd.concat([prescriptions, insert], ignore_INDEX=True)
prescriptions = prescriptions.iloc[:,::-1]
prescriptions.to_csv("urgency_prescriptions_2.csv", INDEX = False,sep = ';')
```

Anexo 3 | Povoamento do Data Warehouse com script SQL

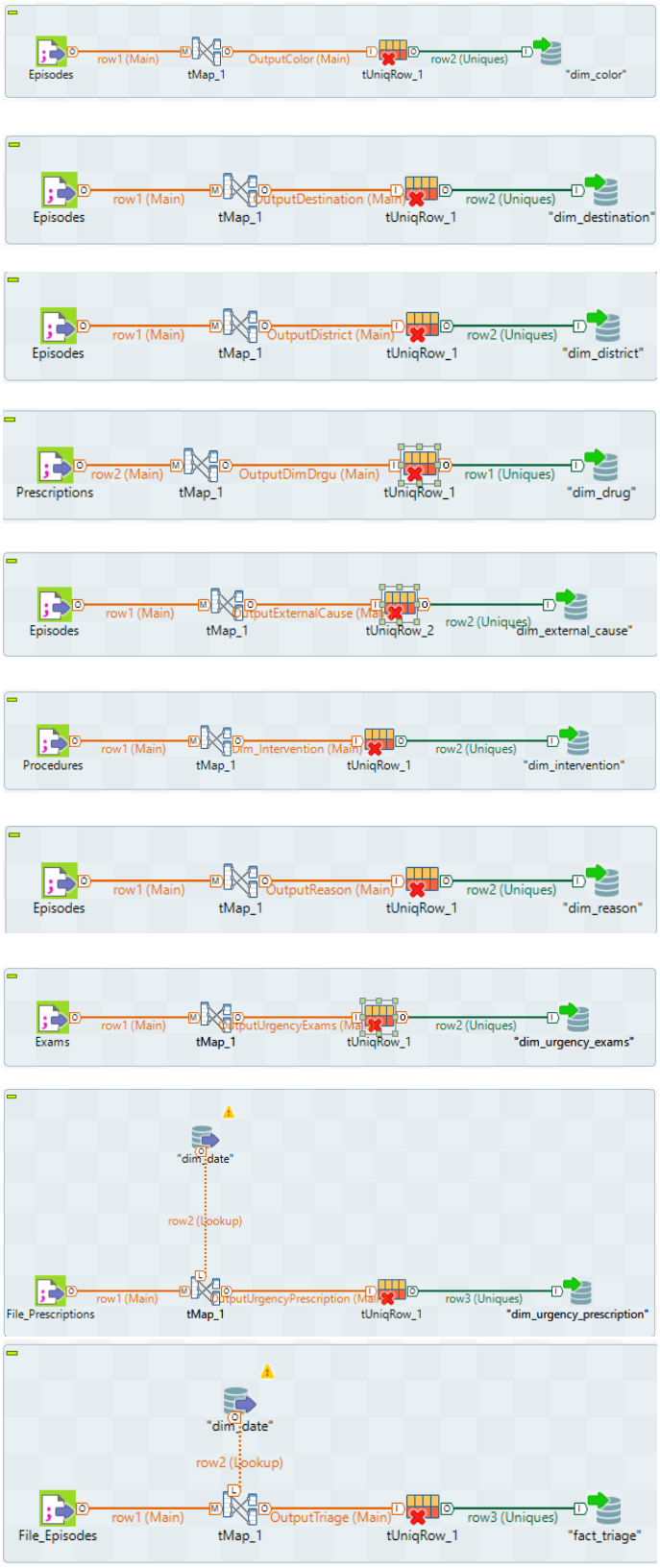
```
-- SCHEMA Urgency
USE Urgency;

-- -----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_DRUG
-- -----------------------------------------------------------------------------
INSERT INTO Dim_Drug (Cod_Drug,Description)
SELECT DISTINCT COD_DRUG, DESC_DRUG FROM `dados`.`urgency_prescriptions`;

-- -----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_DATE
-- -----------------------------------------------------------------------------
DROP TEMPORARY TABLE IF EXISTS Dates;
CREATE TEMPORARY TABLE Dates (`data` DATETIME) ENGINE=MEMORY;
INSERT INTO Dates SELECT STR_TO_DATE(DT_PRESCRIPTION,"%Y/%m/%d %T") FROM `dados`.`urgency_prescriptions`;
INSERT INTO Dates SELECT STR_TO_DATE(DATE_OF_BIRTH,"%Y/%m/%d %T") FROM `dados`.`urgency_episodes`;
INSERT INTO Dates SELECT STR_TO_DATE(DT_ADMITION_URG,"%Y/%m/%d %T") FROM `dados`.`urgency_episodes`;
INSERT INTO Dates SELECT STR_TO_DATE(DT_ADMITION_TRAIGE,"%Y/%m/%d %T") FROM `dados`.`urgency_episodes`;
INSERT INTO Dates SELECT STR_TO_DATE(DT_DIAGNOSIS,"%Y/%m/%d %T") FROM `dados`.`urgency_episodes`;
INSERT INTO Dates SELECT STR_TO_DATE(DT_DISCHARGE,"%Y/%m/%d %T") FROM `dados`.`urgency_episodes`;
INSERT INTO Dates SELECT STR_TO_DATE(DT_PRESCRIPTION,"%Y/%m/%d %T") FROM `dados`.`urgency_procedures`;
INSERT INTO Dates SELECT STR_TO_DATE(DT_BEGIN,"%Y/%m/%d %T") FROM `dados`.`urgency_procedures`;

-- Inserir na Tabela Dim_Date
INSERT INTO Dim_Date(Date)
SELECT DISTINCT data FROM Dates;

-- Eliminar TEMPORARY TABLE
DROP TEMPORARY TABLE Dates;

-- -----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_URGENCY_PRESCRIPTION
-- Dividido em vários slots para conseguir correr
-- -----------------------------------------------------------------------------
INSERT INTO Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION,d.idDate
FROM `dados`.`urgency_prescriptions` uP
LEFT JOIN Dim_Date d ON d.Date = STR_TO_DATE(uP.DT_PRESCRIPTION,"%Y/%m/%d %T")
WHERE `COD_PRESCRIPTION`>= 16369810 AND `COD_PRESCRIPTION` < 16420000;

INSERT INTO Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION, d.idDate
FROM `dados`.`urgency_prescriptions` uP
INNER JOIN Dim_Date d ON d.Date=uP.DT_PRESCRIPTION
WHERE `COD_PRESCRIPTION` >= 16420000 AND `COD_PRESCRIPTION` < 16507000;

INSERT INTO Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION, d.idDate
FROM `dados`.`urgency_prescriptions` uP
LEFT JOIN Dim_Date d ON d.Date=uP.DT_PRESCRIPTION
WHERE `COD_PRESCRIPTION` >= 16507000 AND `COD_PRESCRIPTION` < 16545000;

INSERT INTO Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION, d.idDate
FROM `dados`.`urgency_prescriptions` uP
LEFT JOIN Dim_Date d ON d.Date=uP.DT_PRESCRIPTION
WHERE `COD_PRESCRIPTION` >= 16545000 AND `COD_PRESCRIPTION` < 16584700;

INSERT INTO Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION, d.idDate
FROM `dados`.`urgency_prescriptions` uP
LEFT JOIN Dim_Date d ON d.Date=uP.DT_PRESCRIPTION
WHERE `COD_PRESCRIPTION` >= 16584700 AND `COD_PRESCRIPTION` < 16624000;

INSERT INTO Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION, d.idDate
FROM `dados`.`urgency_prescriptions` uP
LEFT JOIN Dim_Date d ON d.Date=uP.DT_PRESCRIPTION
WHERE `COD_PRESCRIPTION`>= 16624000 AND `COD_PRESCRIPTION` < 16655000;

INSERT INTO Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION, d.idDate
FROM `dados`.`urgency_prescriptions` uP
LEFT JOIN Dim_Date d ON d.Date=uP.DT_PRESCRIPTION
WHERE `COD_PRESCRIPTION`>= 16655000 AND `COD_PRESCRIPTION` < 16694000;
```

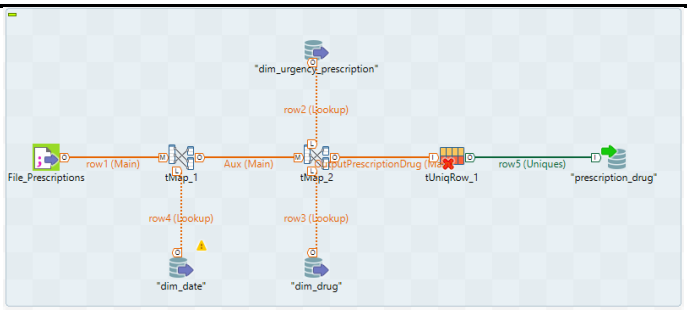
```
INSERT INTO Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
SELECT DISTINCT uP.COD_PRESCRIPTION, uP.ID_PROF_PRESCRIPTION, d.idDate
FROM `dados`.`urgency_prescriptions` uP
LEFT JOIN Dim_Date d ON d.Date=uP.DT_PRESCRIPTION
WHERE `COD_PRESCRIPTION`>= 16694000 AND `COD_PRESCRIPTION` <= 17235985;

-- ----------------------------------------------------------------------------
-- POVOAMENTO DE PRESCRIPTION_DRUG
-- ----------------------------------------------------------------------------
INSERT INTO Prescription_Drug(Urgency_Prescription,Drug,Quantity)
SELECT DISTINCT uP.idUrgency_Prescription,D.idDrug,dUP.QT
FROM `dados`.`urgency_prescriptions` dUP
INNER JOIN Dim_Urgency_Prescription uP ON uP.Cod_Prescription=dUP.COD_PRESCRIPTION AND
dUP.ID_PROF_PRESCRIPTION=uP.Prof_Prescription
INNER JOIN Dim_Drug D ON D.Cod_Drug=dUP.COD_DRUG AND D.Description=dUP.DESC_DRUG;

-- ----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_DISTRICT
-- ----------------------------------------------------------------------------
INSERT INTO Dim_District (District)
SELECT DISTINCT DISTRICT FROM `dados`.`urgency_episodes`;

-- ----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_PATIENT
-- ----------------------------------------------------------------------------
INSERT INTO Dim_Patient(Sex,FK_Date_Of_Birth,FK_District)
SELECT uE.SEX,d.idDate,di.idDistrict
FROM `dados`.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DATE_OF_BIRTH
LEFT JOIN Dim_District di ON di.District=uE.DISTRICT;

-- ----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_URGENCY_EXAMS
-- ----------------------------------------------------------------------------
INSERT INTO Dim_Urgency_Exams(Num_Exame,Description)
SELECT DISTINCT NUM_EXAM,DESC_EXAM FROM `dados`.`urgency_exams`  ;

-- ----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_EXTERNAL_CAUSE
-- ----------------------------------------------------------------------------
INSERT INTO Dim_External_Cause
SELECT DISTINCT ID_EXT_CAUSE,DESC_EXTERNAL_CAUSE FROM `dados`.`urgency_episodes`;

-- ----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_COLOR
-- ----------------------------------------------------------------------------
INSERT INTO Dim_Color
SELECT DISTINCT ID_COLOR, DESC_COLOR FROM `dados`.`urgency_episodes`;

-- ----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_LEVEL
-- ----------------------------------------------------------------------------
-- Povoar todos os niveis 1
INSERT INTO dim_level (idLevel,Cod_Level,Description)
SELECT DISTINCT 1 AS idd,level_1_code,level_1_desc FROM `dados`.`icd9_hierarchy`;
-- Povoar todos os niveis 2
INSERT INTO dim_level (idLevel,Cod_Level,Description)
SELECT DISTINCT 2 AS idd,level_2_code,level_2_desc FROM `dados`.`icd9_hierarchy`;
-- Povoar todos os niveis 3
INSERT INTO dim_level (idLevel,Cod_Level,Description)
SELECT DISTINCT 3 AS idd,level_3_code,level_3_desc FROM `dados`.`icd9_hierarchy`;
-- Povoar todos os niveis 4
INSERT INTO dim_level (idLevel,Cod_Level,Description)
SELECT DISTINCT 4 AS idd,level_4_code,level_4_desc FROM `dados`.`icd9_hierarchy`;
-- Povoar todos os niveis 5
INSERT INTO dim_level (idLevel,Cod_Level,Description)
SELECT DISTINCT 5 AS idd,level_5_code,level_5_desc FROM `dados`.`icd9_hierarchy`;

-- ----------------------------------------------------------------------------
-- POVOAMENTO DE DIM_INFO
-- ----------------------------------------------------------------------------

-- Povoar Dim_Info ONDE o Código contém "E"
INSERT INTO dim_info (cod_diagnosis,description,FK_LevelID,FK_LevelCod)
SELECT DISTINCT a1.COD_DIAGNOSIS, a1.DIAGNOSIS, a2.idLevel, a2.Cod_Level
FROM `dados`.`urgency_episodes` a1
INNER JOIN dim_level a2 ON a2.Cod_Level = 'E'
```

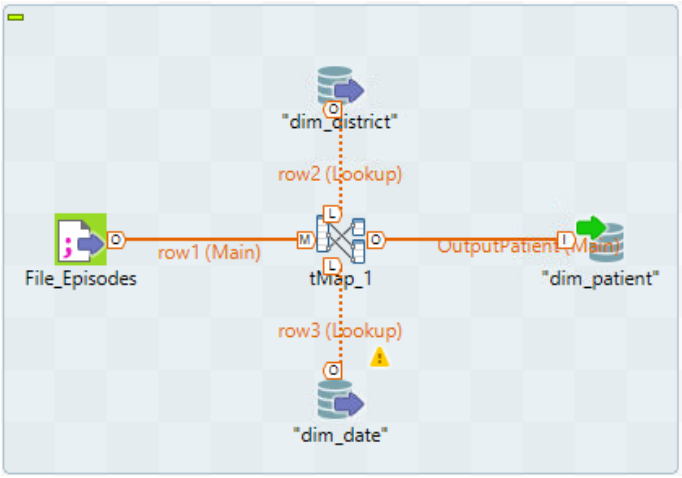
```
WHERE INSTR(a1.COD_DIAGNOSIS,'E') > 0;

-- Povoar DIM_Info ONDE o Código contém "V"
INSERT INTO dim_info (cod_diagnosis,description,FK_LevelID,FK_LevelCod)
SELECT DISTINCT a1.COD_DIAGNOSIS, a1.DIAGNOSIS, a2.idLevel, a2.Cod_Level
FROM `dados`.`urgency_episodes` a1
INNER JOIN dim_level a2 ON a2.Cod_Level = 'V'
WHERE INSTR(a1.COD_DIAGNOSIS,'V') > 0;

-- Povoar DIM_Info ONDE o Código é um intervalo de valores
INSERT INTO dim_info (cod_diagnosis,description,FK_LevelID,FK_LevelCod)
SELECT DISTINCT a1.COD_DIAGNOSIS, a1.DIAGNOSIS, a2.idLevel, a2.Cod_Level
FROM `dados`.`urgency_episodes` a1
INNER JOIN dim_level a2 ON ( (CONVERT(SUBSTRING_INDEX(a2.cod_level,'-',1),DOUBLE) <=
CONVERT(a1.COD_DIAGNOSIS,DOUBLE)) AND ((CONVERT(SUBSTRING_INDEX(a2.cod_level,'-',-1),DOUBLE)+1) >
CONVERT(a1.COD_DIAGNOSIS,DOUBLE)) AND a2.idLevel = 1)
WHERE ( (INSTR(a1.COD_DIAGNOSIS,'V') = 0) AND (INSTR(a1.COD_DIAGNOSIS,'E') = 0) AND (INSTR( a2.Cod_Level,'V') =
0) AND (INSTR( a2.Cod_Level,'E') = 0));

-- ------------------------------------------------------------------------------
-- POVOAMENTO DA TABELA DIM_REASON
-- ------------------------------------------------------------------------------
INSERT INTO Dim_Reason
SELECT DISTINCT ID_REASON,DESC_REASON FROM `dados`.`urgency_episodes`;

-- ------------------------------------------------------------------------------
-- POVOAMENTO DA TABELA DIM_DESTINATION
-- ------------------------------------------------------------------------------
INSERT INTO Dim_Destination
SELECT DISTINCT ID_DESTINATION,DESC_DESTINATION FROM `dados`.`urgency_episodes`;

-- ------------------------------------------------------------------------------
-- POVOAMENTO DA TABELA DE FACTOS - FACT_DIAGNOSIS
-- ------------------------------------------------------------------------------
INSERT INTO Fact_Diagnosis
(Urg_Episode,Prof_Diagnosis,Prof_Discharge,FK_Date_Diagnosis,FK_Destination,FK_Date_Discharge,FK_Reason,FK_Info)
SELECT DISTINCT a1.URG_EPISODE, a1.ID_PROF_DIAGNOSIS, a1.ID_PROF_DISCHARGE,
a2.idDate,a3.idDestination,a4.idDate,a5.idReason,a6.idInfo
FROM `dados`.`urgency_episodes` a1
INNER JOIN Dim_Date a2 ON STR_TO_DATE(a1.DT_DIAGNOSIS,"%Y/%m/%d %T") = a2.Date
INNER JOIN Dim_Destination a3 ON a1.DESC_DESTINATION = a3.Description
INNER JOIN Dim_Date a4 ON STR_TO_DATE(a1.DT_DISCHARGE,"%Y/%m/%d %T") = a4.Date
INNER JOIN Dim_Reason a5 ON a1.DESC_REASON = a5.Description
INNER JOIN Dim_Info a6 ON a1.COD_DIAGNOSIS = a6.Cod_Diagnosis AND a1.DIAGNOSIS = a6.Description;

-- ------------------------------------------------------------------------------
-- POVOAMENTO DA TABELA DE FACTOS - FACT_TRIAGE
-- Dividido em vários slots para conseguir correr
-- ------------------------------------------------------------------------------
INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE <= 1789698;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1789698 AND uE.URG_EPISODE <= 1797718;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1797718 AND uE.URG_EPISODE <= 1805634;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1805634 AND uE.URG_EPISODE <= 1813337;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
```

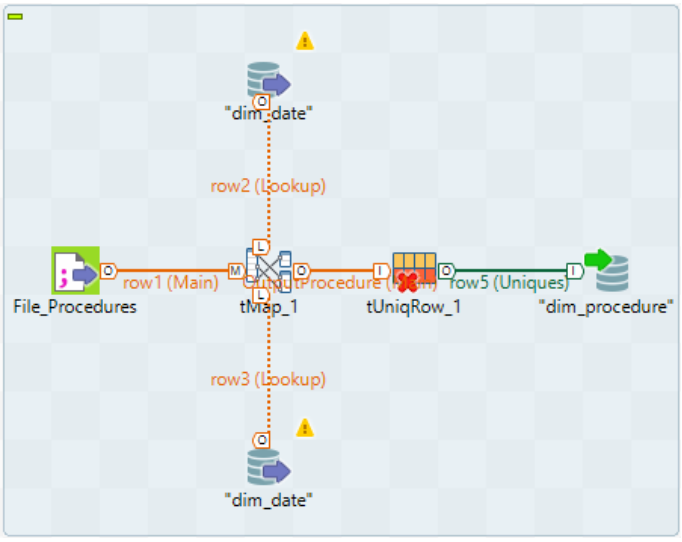
```
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1813337 AND uE.URG_EPISODE <= 1821197;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1821197 AND uE.URG_EPISODE <= 1829098;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1829098 AND uE.URG_EPISODE <= 1836948;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1836948 AND uE.URG_EPISODE <= 1844779;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1844779 AND uE.URG_EPISODE <= 1860174;

INSERT INTO Fact_Triage(Urg_Episode, Prof_Triagem, Pain_Scale, FK_Date_Admission, FK_Color)
SELECT DISTINCT uE.URG_EPISODE, uE.ID_PROF_TRIAGE, uE.PAIN_SCALE, d.idDate, c.idColor
FROM dados.`urgency_episodes` uE
LEFT JOIN Dim_Date d ON d.Date=uE.DT_ADMITION_TRAIGE
LEFT JOIN Dim_Color c ON c.idColor = uE.ID_COLOR
WHERE uE.URG_EPISODE > 1860174 AND uE.URG_EPISODE <= 1883820;

-- ------------------------------------------------------------------------------
-- POVOAMENTO DE DIM_INTERVENTION
-- ------------------------------------------------------------------------------
INSERT INTO Dim_Intervention
SELECT DISTINCT ID_INTERVENTION, DESC_INTERVENTION FROM `dados`.`urgency_procedures`;

-- ------------------------------------------------------------------------------
-- POVOAMENTO DE DIM_PROCEDURE
-- ------------------------------------------------------------------------------
INSERT INTO Dim_Procedure (idPrescription, Prof_Procedure, Prof_Cancel, Canceled, FK_Date_Prescription,
FK_Date_Begin, FK_Intervention)
SELECT a1.ID_PRESCRIPTION, a1.ID_PROFESSIONAL, a1.ID_PROFESSIONAL_CANCEL, a1.DT_CANCEL, a2.idDate, a3.idDate,
a4.idIntervention
FROM `dados`.`urgency_procedures` a1
INNER JOIN Dim_Date a2 ON STR_TO_DATE(a1.DT_PRESCRIPTION,"%Y/%m/%d %T") = a2.Date
INNER JOIN Dim_Date a3 ON STR_TO_DATE(a1.DT_BEGIN,"%Y/%m/%d %T") = a3.Date
INNER JOIN Dim_Intervention a4 ON a1.ID_INTERVENTION = a4.idIntervention;
```

Anexo 4 | Povoamento do *Data Warehouse* através do *Talend*




"dim_urgency_prescription"
row2 (Lookup)
File_Prescriptions
row1 (Main)
tMap_1
Aux (Main)
tMap_2
tUniqRow_1
row5 (Uniques)
"prescription_drug"
row4 (Lookup)
row3 (Lookup)
"dim_date"
"dim_drug"


"dim_district"
row2 (Lookup)
File_Episodes
row1 (Main)
tMap_1
"dim_patient"
row3 (Lookup)
"dim_date"


"dim_date"
row2 (Lookup)
File_Procedures
row1 (Main)
tMap_1
tUniqRow_1
row5 (Uniques)
"dim_procedure"
row3 (Lookup)
"dim_date"

Anexo 5 | Criação da área de retenção

```
-- MySQL Workbench Forward Engeneering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO
,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema Urgency_Retencao
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `Urgency_Retencao` DEFAULT CHARACTER SET utf8 ;
USE `Urgency_Retencao`;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Color`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Color` (
  `idColor` INT NOT NULL,
  `Description` VARCHAR(8) NOT NULL,
  PRIMARY KEY (`idColor`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Reason`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Reason` (
  `idReason` INT NOT NULL,
  `Description` VARCHAR(33) NOT NULL,
  PRIMARY KEY (`idReason`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Destination`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Destination` (
  `idDestination` INT NOT NULL,
  `Description` VARCHAR(43) NOT NULL,
  PRIMARY KEY (`idDestination`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Level`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Level` (
  `idLevel` INT NOT NULL,
  `Cod_Level` VARCHAR(15) NOT NULL,
  `Description` VARCHAR(250) NOT NULL,
  PRIMARY KEY (`idLevel`, `Cod_Level`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Info`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Info` (
  `idInfo` INT NOT NULL AUTO_INCREMENT,
  `Cod_Diagnosis` VARCHAR(45) NOT NULL,
  `Description` VARCHAR(250) NOT NULL,
  `FK_LevelID` INT NOT NULL,
  `FK_LevelCod` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`idInfo`),
  INDEX `fk_Dim_Info_Dim_Level_idx` (`FK_LevelID` ASC, `FK_LevelCod` ASC) VISIBLE,
  CONSTRAINT `fk_Dim_Info_Dim_Level`
    FOREIGN KEY (`FK_LevelID` , `FK_LevelCod`)
    REFERENCES `Urgency_Retencao`.`Dim_Level` (`idLevel` , `Cod_Level`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Date`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Date` (
  `idDate` INT NOT NULL AUTO_INCREMENT,
  `Date` DATETIME NOT NULL,
  PRIMARY KEY (`idDate`))
```

```
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Fact_Diagnosis`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Fact_Diagnosis` (
  `Urg_Episode` INT NOT NULL,
  `Prof_Diagnosis` INT NOT NULL,
  `Prof_Discharge` INT NOT NULL,
  `FK_Date_Diagnosis` INT NOT NULL,
  `FK_Destination` INT NOT NULL,
  `FK_Date_Discharge` INT NOT NULL,
  `FK_Reason` INT NOT NULL,
  `FK_Info` INT NOT NULL,
  PRIMARY KEY (`Urg_Episode`),
  INDEX `fk_Fact_Diagnosis_Dim_Info1_idx` (`FK_Info` ASC) VISIBLE,
  INDEX `fk_Fact_Diagnosis_Dim_Reason1_idx` (`FK_Reason` ASC) VISIBLE,
  INDEX `fk_Fact_Diagnosis_Dim_Destination1_idx` (`FK_Destination` ASC) VISIBLE,
  INDEX `fk_Fact_Diagnosis_Dim_Date1_idx` (`FK_Date_Diagnosis` ASC) VISIBLE,
  INDEX `fk_Fact_Diagnosis_Dim_Date2_idx` (`FK_Date_Discharge` ASC) VISIBLE,
  CONSTRAINT `fk_Fact_Diagnosis_Dim_Info1`
    FOREIGN KEY (`FK_Info`)
    REFERENCES `Urgency_Retencao`.`Dim_Info` (`idInfo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Diagnosis_Dim_Reason1`
    FOREIGN KEY (`FK_Reason`)
    REFERENCES `Urgency_Retencao`.`Dim_Reason` (`idReason`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Diagnosis_Dim_Destination1`
    FOREIGN KEY (`FK_Destination`)
    REFERENCES `Urgency_Retencao`.`Dim_Destination` (`idDestination`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Diagnosis_Dim_Date1`
    FOREIGN KEY (`FK_Date_Diagnosis`)
    REFERENCES `Urgency_Retencao`.`Dim_Date` (`idDate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Diagnosis_Dim_Date2`
    FOREIGN KEY (`FK_Date_Discharge`)
    REFERENCES `Urgency_Retencao`.`Dim_Date` (`idDate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Fact_Triage`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Fact_Triage` (
  `Urg_Episode` INT NOT NULL,
  `Prof_Triagem` INT NOT NULL,
  `Pain_Scale` INT NOT NULL,
  `FK_Date_Admission` INT NOT NULL,
  `FK_Color` INT NOT NULL,
  PRIMARY KEY (`Urg_Episode`),
  INDEX `fk_Fact_Triage_Dim_Date1_idx` (`FK_Date_Admission` ASC) VISIBLE,
  INDEX `fk_Fact_Triage_Dim_Color1_idx` (`FK_Color` ASC) VISIBLE,
  CONSTRAINT `fk_Fact_Triage_Dim_Date1`
    FOREIGN KEY (`FK_Date_Admission`)
    REFERENCES `Urgency_Retencao`.`Dim_Date` (`idDate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Triage_Dim_Color1`
    FOREIGN KEY (`FK_Color`)
    REFERENCES `Urgency_Retencao`.`Dim_Color` (`idColor`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Intervention`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Intervention` (
  `idIntervention` INT NOT NULL,
  `Description` VARCHAR(155) NOT NULL,
  PRIMARY KEY (`idIntervention`))
```

```
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_External_Cause`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_External_Cause` (
  `idExternal_Cause` INT NOT NULL,
  `Description` VARCHAR(24) NOT NULL,
  PRIMARY KEY (`idExternal_Cause`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Urgency_Exams`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Urgency_Exams` (
  `idUrgency_Exams` INT NOT NULL AUTO_INCREMENT,
  `Num_Exame` VARCHAR(23) NOT NULL,
  `Description` VARCHAR(104) NOT NULL,
  PRIMARY KEY (`idUrgency_Exams`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Procedure`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Procedure` (
  `idPrescription` INT NOT NULL,
  `Prof_Procedure` INT NOT NULL,
  `Prof_Cancel` INT NOT NULL,
  `Canceled` INT NOT NULL,
  `FK_Date_Prescription` INT NOT NULL,
  `FK_Date_Begin` INT NOT NULL,
  `FK_Intervention` INT NOT NULL,
  PRIMARY KEY (`idPrescription`),
  INDEX `fk_Dim_Procedure_Dim_Intervention1_idx` (`FK_Intervention` ASC) VISIBLE,
  INDEX `fk_Dim_Procedure_Dim_Date1_idx` (`FK_Date_Prescription` ASC) VISIBLE,
  INDEX `fk_Dim_Procedure_Dim_Date2_idx` (`FK_Date_Begin` ASC) VISIBLE,
  CONSTRAINT `fk_Dim_Procedure_Dim_Intervention1`
    FOREIGN KEY (`FK_Intervention`)
    REFERENCES `Urgency_Retencao`.`Dim_Intervention` (`idIntervention`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Dim_Procedure_Dim_Date1`
    FOREIGN KEY (`FK_Date_Prescription`)
    REFERENCES `Urgency_Retencao`.`Dim_Date` (`idDate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Dim_Procedure_Dim_Date2`
    FOREIGN KEY (`FK_Date_Begin`)
    REFERENCES `Urgency_Retencao`.`Dim_Date` (`idDate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_District`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_District` (
  `idDistrict` INT NOT NULL AUTO_INCREMENT,
  `District` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`idDistrict`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Patient`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Patient` (
  `idPatient` INT NOT NULL AUTO_INCREMENT,
  `Sex` VARCHAR(1) NOT NULL,
  `FK_Date_Of_Birth` INT NOT NULL,
  `FK_District` INT NOT NULL,
  PRIMARY KEY (`idPatient`),
  INDEX `fk_Dim_Patient_Dim_Date1_idx` (`FK_Date_Of_Birth` ASC) VISIBLE,
  INDEX `fk_Dim_Patient_Dim_District1_idx` (`FK_District` ASC) VISIBLE,
  CONSTRAINT `fk_Dim_Patient_Dim_Date1`
    FOREIGN KEY (`FK_Date_Of_Birth`)
    REFERENCES `Urgency_Retencao`.`Dim_Date` (`idDate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
```

```
  CONSTRAINT `fk_Dim_Patient_Dim_District1`
    FOREIGN KEY (`FK_District`)
    REFERENCES `Urgency_Retencao`.`Dim_District` (`idDistrict`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Drug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Drug` (
  `idDrug` INT NOT NULL AUTO_INCREMENT,
  `Cod_Drug` INT NOT NULL,
  `Description` VARCHAR(227) NOT NULL,
  PRIMARY KEY (`idDrug`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Dim_Urgency_Prescription`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Dim_Urgency_Prescription` (
  `idUrgency_Prescription` INT NOT NULL AUTO_INCREMENT,
  `Cod_Prescription` INT NOT NULL,
  `Prof_Prescription` INT NOT NULL,
  `FK_Date_Prescription` INT NOT NULL,
  PRIMARY KEY (`idUrgency_Prescription`),
  INDEX `fk_Dim_Urgency_Prescription_Dim_Date1_idx` (`FK_Date_Prescription` ASC) VISIBLE,
  CONSTRAINT `fk_Dim_Urgency_Prescription_Dim_Date1`
    FOREIGN KEY (`FK_Date_Prescription`)
    REFERENCES `Urgency_Retencao`.`Dim_Date` (`idDate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Prescription_Drug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Prescription_Drug` (
  `Urgency_Prescription` INT NOT NULL,
  `Drug` INT NOT NULL,
  `Quantity` INT NOT NULL,
  PRIMARY KEY (`Urgency_Prescription`, `Drug`, `Quantity`),
  INDEX `fk_Dim_Urgency_Prescription_has_Dim_Drug_Dim_Drug1_idx` (`Drug` ASC) VISIBLE,
  INDEX `fk_Dim_Urgency_Prescription_has_Dim_Drug_Dim_Urgency_Prescr_idx` (`Urgency_Prescription` ASC)
VISIBLE,
  CONSTRAINT `fk_Dim_Urgency_Prescription_has_Dim_Drug_Dim_Urgency_Prescrip1`
    FOREIGN KEY (`Urgency_Prescription`)
    REFERENCES `Urgency_Retencao`.`Dim_Urgency_Prescription` (`idUrgency_Prescription`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Dim_Urgency_Prescription_has_Dim_Drug_Dim_Drug1`
    FOREIGN KEY (`Drug`)
    REFERENCES `Urgency_Retencao`.`Dim_Drug` (`idDrug`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Urgency_Retencao`.`Fact_Urgency_Episodes`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Urgency_Retencao`.`Fact_Urgency_Episodes` (
  `Urg_Episode` INT NOT NULL,
  `Prof_Admission` INT NOT NULL,
  `FK_Patient` INT NOT NULL,
  `FK_Date_Admission` INT NOT NULL,
  `FK_External_Cause` INT NOT NULL,
  `FK_Urgency_Exams` INT NOT NULL,
  `FK_Procedure` INT NOT NULL,
  `FK_Urgency_Prescription` INT NOT NULL,
  PRIMARY KEY (`Urg_Episode`, `FK_Urgency_Exams`, `FK_Procedure`, `FK_Urgency_Prescription`),
  INDEX `fk_Fact_Urgency_Episodes_Dim_Patient1_idx` (`FK_Patient` ASC) VISIBLE,
  INDEX `fk_Fact_Urgency_Episodes_Dim_Date1_idx` (`FK_Date_Admission` ASC) VISIBLE,
  INDEX `fk_Fact_Urgency_Episodes_Dim_External_Cause1_idx` (`FK_External_Cause` ASC) VISIBLE,
  INDEX `fk_Fact_Urgency_Episodes_Dim_Urgency_Exams1_idx` (`FK_Urgency_Exams` ASC) VISIBLE,
  INDEX `fk_Fact_Urgency_Episodes_Dim_Procedure1_idx` (`FK_Procedure` ASC) VISIBLE,
```

```
  INDEX `fk_Fact_Urgency_Episodes_Dim_Urgency_Prescription1_idx` (`FK_Urgency_Prescription` ASC)
VISIBLE,
  CONSTRAINT `fk_Fact_Urgency_Episodes_Dim_Patient1`
    FOREIGN KEY (`FK_Patient`)
    REFERENCES `Urgency_Retencao`.`Dim_Patient` (`idPatient`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Urgency_Episodes_Dim_Date1`
    FOREIGN KEY (`FK_Date_Admission`)
    REFERENCES `Urgency_Retencao`.`Dim_Date` (`idDate`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Urgency_Episodes_Dim_External_Cause1`
    FOREIGN KEY (`FK_External_Cause`)
    REFERENCES `Urgency_Retencao`.`Dim_External_Cause` (`idExternal_Cause`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Urgency_Episodes_Dim_Urgency_Exams1`
    FOREIGN KEY (`FK_Urgency_Exams`)
    REFERENCES `Urgency_Retencao`.`Dim_Urgency_Exams` (`idUrgency_Exams`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Urgency_Episodes_Dim_Procedure1`
    FOREIGN KEY (`FK_Procedure`)
    REFERENCES `Urgency_Retencao`.`Dim_Procedure` (`idPrescription`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Fact_Urgency_Episodes_Dim_Urgency_Prescription1`
    FOREIGN KEY (`FK_Urgency_Prescription`)
    REFERENCES `Urgency_Retencao`.`Dim_Urgency_Prescription` (`idUrgency_Prescription`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Anexo 6 | Inserção dos dados na área de retenção

```
-- -----------------------------------------------------
-- Schema Urgency_Retencao
-- -----------------------------------------------------
USE urgency_retencao;

-- -----------------------------------------------------
-- Inserir Valores de tabelas estáticas
-- -----------------------------------------------------
INSERT INTO dim_level SELECT * FROM urgency.dim_level;

-- -----------------------------------------------------------------------------------------------
-- Procedure para inserir uma nova triagem, aceita valores nulos como entrada em todas as posições
-- exceto código de urgência.
-- -----------------------------------------------------------------------------------------------
DROP PROCEDURE IF EXISTS InserirNovaTriagem;
DELIMITER $$
CREATE PROCEDURE InserirNovaTriagem(UEpisode INT, TProfissional INT, PainScale INT, DataAdmissao
VARCHAR(100),idCor INT,Cor VARCHAR(8))
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0; DECLARE nn_TProfissional INT;
    DECLARE nn_PainScale INT; DECLARE nn_DataAdmissao VARCHAR(100);
    DECLARE nn_idCor INT; DECLARE nn_Cor VARCHAR(8);

    SET nn_TProfissional = IFNULL(TProfissional,-1);
    SET nn_PainScale = IFNULL(PainScale,-1);
    SET nn_DataAdmissao = IFNULL(DataAdmissao,'1111/11/11 00:00:00');
    SET nn_idCor = IFNULL(idCor,-1);
    SET nn_Cor = IFNULL(Cor,'Unknown');

    IF UEpisode IS NULL THEN SET Erro = 1; END IF;
    IF Erro = 1 THEN SELECT 'Número de Episódio de Urgência não pode ser nulo.'; END IF;

    START TRANSACTION;
    INSERT INTO Fact_Triage(Urg_Episode,Prof_Triagem,Pain_Scale,FK_Date_Admission,FK_Color) VALUES
(UEpisode,nn_TProfissional,nn_PainScale,getFKdata(nn_DataAdmissao),getFKcor(nn_idCor,nn_Cor));

END $$
DELIMITER ;

-- -----------------------------------------------------------------------------------------------
-- Procedure para inserir um novo diagnóstico. Aceita valores nulos em todas as posições exceto código
de urgência, código de nivel e descrição de nível
-- -----------------------------------------------------------------------------------------------
DROP PROCEDURE IF EXISTS InserirNovoDiagnostico;
DELIMITER $$
CREATE PROCEDURE InserirNovoDiagnostico(UEpisode INT,DProfissional INT,DchProfissional INT,DataD
VARCHAR(100),Destino INT,DestinoDesc VARCHAR(43),DataDch VARCHAR(100), Reason INT, ReasonDesc
VARCHAR(33), Cod VARCHAR(45), Cod_Desc VARCHAR(250))
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE nn_DProfissional INT; DECLARE nn_DchProfissional INT; DECLARE nn_DataD VARCHAR(100); DECLARE
nn_Destino INT;
    DECLARE nn_DestinoDesc VARCHAR(43); DECLARE nn_DataDch VARCHAR(100); DECLARE nn_Reason INT; DECLARE
nn_ReasonDesc VARCHAR(33);

    IF UEpisode IS NULL THEN SET Erro = 1; END IF;
    IF Erro = 1 THEN SELECT 'Número de Episódio de Urgência não pode ser nulo.'; END IF;

    IF (Cod IS NULL OR Cod_Desc IS NULL) THEN SET Erro = 1; END IF;
    IF Erro = 1 THEN SELECT 'Valor de Código ou Descrição não pode ser nulo.'; END IF;

    SET nn_DProfissional = IFNULL(DProfissional,-1);
    SET nn_DchProfissional = IFNULL(DchProfissional,-1);
    SET nn_DataD = IFNULL(DataD,'1111/11/11 00:00:00');
    SET nn_Destino = IFNULL(Destino,-1);
    SET nn_DestinoDesc = IFNULL(DestinoDesc,"Sem destino indicado.");
    SET nn_DataDch = IFNULL(DataDch,'1111/11/11 00:00:00');
    SET nn_Reason = IFNULL(Reason,-1);
    SET nn_ReasonDesc = IFNULL(ReasonDesc,"Destino de Alta não fornecido.");

    START TRANSACTION;
    INSERT INTO Fact_Diagnosis(Urg_Episode, Prof_Diagnosis, Prof_Discharge, FK_Date_Diagnosis,
FK_Destination, FK_Date_Discharge, FK_Reason, FK_Info) VALUES (Uepisode, nn_DProfissional,
nn_DchProfissional, getFKdata(nn_DataD), getFKdestino(nn_Destino, nn_DestinoDesc),
getFkdata(nn_DataDch),getFKreason(nn_Reason,nn_ReasonDesc), getFKinfo(Cod,Cod_Desc));
```

```
END $$
DELIMITER ;
-- -----------------------------------------------------------------------------------------------
-- Procedure para inserir um novo nivel de diagnóstico e a sua descrição. Valores nulos não tolerados
-- -----------------------------------------------------------------------------------------------
DELIMITER
CREATE PROCEDURE InserirNovoDimInfo(Codigo VARCHAR(45), Descricao VARCHAR(250))
BEGIN
    DECLARE Id INT;
    DECLARE Cod VARCHAR(15);

    SET Id = 1;
    IF ( INSTR(Codigo,'E') > 0 ) THEN SET Cod = 'E'; END IF;
    IF ( INSTR(Codigo,'V') > 0 ) THEN SET Cod = 'V'; END IF;
    IF ( (INSTR(Codigo,'V') = 0) AND (INSTR(Codigo,'E') = 0) )
       THEN SET Cod = (SELECT l1.Cod_Level FROM Dim_Level l1
                       WHERE (l1.idLevel = 1) AND ((CONVERT(SUBSTRING_INDEX(l1.cod_level,'-',1), DOUBLE)
<= CONVERT(Codigo,DOUBLE)) AND
                       ((CONVERT(SUBSTRING_INDEX(l1.cod_level,'-',-1),DOUBLE)+1) >
CONVERT(Codigo,DOUBLE) ) ) );
    END IF;

    INSERT INTO Dim_Info(Cod_Diagnosis,Description,FK_LevelID,FK_LevelCod) VALUES
(Codigo,Descricao,Id,Cod);

END $$
DELIMITER ;

-- -----------------------------------------------------
-- Procedure para inserir uma nova Urgency Prescription
-- -----------------------------------------------------
DROP PROCEDURE IF EXISTS InsereUrgencyPrescription;
DELIMITER $$
CREATE PROCEDURE InsereUrgencyPrescription(Cod_Prescription INT, Prof_Prescription INT,
Date_Prescription VARCHAR(100))
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE nn_ProfPrescription INT; DECLARE nn_DatePrescription VARCHAR(100);DECLARE nn_CodPrescription
INT;

    SET nn_CodPrescription = IFNULL(Cod_Prescription,-1);
    SET nn_ProfPrescription = IFNULL(Prof_Prescription,-1);
    SET nn_DatePrescription = IFNULL(Date_Prescription,'1111/11/11 00:00:00');

START TRANSACTION;
   INSERT INTO Dim_Urgency_Prescription(Cod_Prescription, Prof_Prescription, FK_Date_Prescription)
VALUES (nn_CodPrescription, nn_ProfPrescription, getFKdata(nn_DatePrescription));
END $$
DELIMITER ;

CALL InsereUrgencyPrescription(null,null,null);

-- -----------------------------------------------------------------------------
-- Procedure para inserir novas entradas na tabela N:N. Não permite valores nulos
-- -----------------------------------------------------------------------------
DROP PROCEDURE IF EXISTS InsereRelacaoPrescriptionDrug;
DELIMITER $$
CREATE PROCEDURE InsereRelacaoPrescriptionDrug(Cod_Prescription INT,Cod_Drug INT,Drug_Description
VARCHAR(227),Quantity INT)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0; DECLARE nn_CodDrug INT;
    DECLARE nn_DrugDescription VARCHAR(227); DECLARE nn_Quantity INT; DECLARE fk_DUP INT;
    SET nn_CodDrug = IFNULL(Cod_Drug,-1);
    SET nn_DrugDescription = IFNULL(Drug_Description,"Sem Medicação");
    SET nn_Quantity = IFNULL(Quantity,0);

    IF Cod_Prescription IS NULL THEN SET Erro = 1; END IF;
    IF Erro = 1 THEN SELECT 'Código de Prescrição não pode ser nulo.'; END IF;
    IF Cod_Prescription NOT IN (SELECT a1.Cod_Prescription FROM Dim_Urgency_Prescription a1) THEN SET
Erro = 1; END IF;
    IF Erro = 1 THEN SELECT 'Não existe Episódio de Urgência com o código de prescrição fornecido.
Inválido.'; END IF;
    SET fk_DUP = (SELECT a1.idUrgency_Prescription FROM Dim_Urgency_Prescription a1 WHERE
a1.Cod_Prescription = Cod_Prescription);

    START TRANSACTION;
    INSERT INTO Prescription_Drug(Urgency_Prescription, Drug,Quantity) VALUES (fk_DUP,
getFKdrug(nn_CodDrug, nn_DrugDescription), nn_Quantity);
```

```
END $$
DELIMITER ;
DROP PROCEDURE IF EXISTS InserirProcedimento;
DELIMITER $$
CREATE PROCEDURE InserirProcedimento(idPrescription INT,profP INT,profC INT,canceled INT,dateP
VARCHAR(100), dateB VARCHAR(100),idIntervention INT,intervention VARCHAR(155))
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0; DECLARE nn_idPrescription INT;
    DECLARE nn_profP INT; DECLARE nn_profC INT;
    DECLARE nn_canceled INT; DECLARE nn_dateP VARCHAR(100);
    DECLARE nn_dateB VARCHAR(100); DECLARE nn_idIntervention INT;
    DECLARE nn_intervention VARCHAR(155);

    SET nn_idPrescription = IFNULL(idPrescription,-1);
    SET nn_profP = IFNULL(profP,-1);
    SET nn_profC = IFNULL(profC,-1);
    SET nn_canceled = IFNULL(canceled,0);
    SET nn_dateP = IFNULL(dateP,'1111/11/11 00:00:00');
    SET nn_dateB = IFNULL(dateB,'1111/11/11 00:00:00');
    SET nn_idIntervention = IFNULL(idIntervention,'-1');
    SET nn_intervention = IFNULL(intervention,'Sem intervenção');

    START TRANSACTION;
    INSERT INTO Dim_Procedure(idPrescription, Prof_Procedure, Prof_Cancel, Canceled,
FK_Date_Prescription, FK_Date_Begin, FK_intervention) VALUES (nn_idPrescription, nn_profP,nn_profC,
nn_canceled, getFKdata(nn_dateP), getFKdata(nn_dateB), getFKintervention(nn_idIntervention,
nn_intervention));
END $$
DELIMITER ;
CALL urgency_retencao.InserirProcedimento(null, null, null, null, null, null, null, null);

-- -----------------------------------------------------
-- Insere pacientes , 1 episódio de urgência = 1 doente
-- -----------------------------------------------------
DROP PROCEDURE IF EXISTS InserirNovoPaciente;
DELIMITER $$
CREATE PROCEDURE InserirNovoPaciente(Sexo VARCHAR(1), DataNascimento VARCHAR(100), Distrito VARCHAR(20))
BEGIN
    START TRANSACTION;
    INSERT INTO Dim_Patient(Sex,FK_Date_Of_Birth,FK_District) VALUES (Sexo, (SELECT
getFKdata(DataNascimento)), (SELECT getFKdistrito(Distrito)));

END $$
DELIMITER ;

-- -----------------------------------------------------
-- Insere um novo episódio de urgência
-- -----------------------------------------------------
DROP PROCEDURE IF EXISTS InserirEpisodioUrgencia;
DELIMITER $$
CREATE PROCEDURE InserirEpisodioUrgencia(Urg_Episode INT, Prof_Admission INT, Sex VARCHAR(1),
Date_Of_Birth VARCHAR(100), District VARCHAR(20), Date_Admission VARCHAR(100), idExtC INT,ExtC
VARCHAR(24), Num_Exame VARCHAR(23), Exam_Description VARCHAR(104), idPrescription INT, idProcedure INT)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0; DECLARE nn_ProfA INT;
    DECLARE nn_Sex VARCHAR(1); DECLARE nn_DoB VARCHAR(100); DECLARE nn_District VARCHAR(20);
    DECLARE nn_DateAdd VARCHAR(100);
    DECLARE nn_idExtC INT; DECLARE ExtC VARCHAR(24);
    DECLARE nn_NumExame VARCHAR(23); DECLARE nn_ExamDescription VARCHAR(104);
    DECLARE nn_Prescription INT; DECLARE nn_Procedure INT; DECLARE pp INT;

    IF Urg_Episode IS NULL THEN SET Erro = 1; END IF;
    IF Erro = 1 THEN SELECT 'Número de Episódio de Emergência não pode ser nulo.'; END IF;

    SET nn_ProfA = IFNULL(Prof_Admission,-1);
    SET nn_Sex = IFNULL(Sex,'N');S ET nn_DoB = IFNULL(Date_Of_Birth,'1111/11/11 00:00:00');
    SET nn_District = IFNULL(District,'Não Indicado');
    SET nn_DateAdd = IFNULL(Date_Admission,'1111/11/11 00:00:00');
    SET nn_idExtC = IFNULL(idExtC,-1);SET ExtC = IFNULL(ExtC,'Desconhecida');
    SET nn_NumExame = IFNULL(Num_Exame,'Sem Exame');
    SET nn_ExamDescription = IFNULL(Exam_Description,'Sem Exame');
    SET nn_Prescription = IFNULL(idPrescription,-1);SET nn_Procedure = IFNULL(idProcedure,-1);

    IF nn_Prescription NOT IN (SELECT a1.Cod_Prescription FROM Dim_Urgency_Prescription a1) THEN SET
Erro = 1; END IF;
    IF Erro = 1 THEN SELECT 'Não existe Episódio de Urgência com o código de prescrição fornecido.
Inválido.'; END IF;
```

```
    IF nn_Procedure NOT IN (SELECT a1.idPrescription FROM Dim_Procedure a1) THEN SET Erro = 1; END IF;
    IF Erro = 1 THEN SELECT 'Não existe Episódio de Urgência com o código fornecido. Inválido.'; END IF;
    SET pp = (SELECT a1.idUrgency_Prescription FROM Dim_Urgency_Prescription a1 WHERE
a1.Cod_Prescription = nn_Prescription);

    START TRANSACTION;
    INSERT INTO
Fact_Urgency_Episodes(Urg_Episode,Prof_Admission,FK_Patient,FK_Date_Admission,FK_External_Cause,FK_Urgen
cy_Exams,FK_Urgency_Prescription,FK_Procedure) VALUES (Urg_Episode, nn_ProfA, getFKpatient(nn_Sex,
nn_DoB,nn_District), getFKdata(nn_DateAdd), getFKextCause(nn_idExtC,ExtC), getFKexame(nn_NumExame,
nn_ExamDescription), pp,nn_Procedure);
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK da data
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKdata;
DELIMITER $$
CREATE FUNCTION getFKdata ( DataNascimento VARCHAR(100) ) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE dataConvertida DATETIME; DECLARE chave INT;
     SET dataConvertida = STR_TO_DATE(DataNascimento,"%Y/%m/%d %T");

     IF (dataConvertida NOT IN ( SELECT a1.Date FROM Dim_Date a1) ) THEN INSERT INTO Dim_Date (date)
VALUES (dataConvertida); END IF;
     SET chave =  (SELECT a1.idDate FROM Dim_Date a1 WHERE dataConvertida = a1.Date);

     RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK da cor
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKcor;
DELIMITER $$
CREATE FUNCTION getFKcor ( idCor INT,Cor VARCHAR(8) ) RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE chave INT;
   IF (idCor NOT IN ( SELECT a1.idColor FROM Dim_Color a1) AND Cor NOT IN ( SELECT a2.Description FROM
Dim_Color a2)) THEN INSERT INTO Dim_Color(idColor,Description) VALUES (idCor, Cor); END IF;
   IF (Cor IN ( SELECT a1.Description FROM Dim_Color a1) ) THEN SET chave = (SELECT a1.idColor FROM
Dim_Color a1 WHERE a1.Description = Cor); END IF;
   IF (idCor IN (SELECT a1.idColor FROM Dim_Color a1) AND (idCor <> -1)) THEN SET chave = idCor; END IF;

   RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK do destino
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKdestino;
DELIMITER $$
CREATE FUNCTION getFKdestino (Id INT,Destino VARCHAR(43)) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE chave INT;

    IF (Id NOT IN ( SELECT a1.idDestination FROM Dim_Destination a1) AND Destino NOT IN ( SELECT
a2.Description FROM Dim_Destination a2))THEN INSERT INTO Dim_Destination(idDestination,Description)
VALUES (Id,Destino); END IF;
    IF (Destino IN ( SELECT a1.Description FROM Dim_Destination a1) ) THEN SET chave = (SELECT
a1.idDestination FROM Dim_Destination a1 WHERE a1.Description = Destino); END IF;
    IF (Id IN ( SELECT a1.idDestination FROM Dim_Destination a1) AND (Id <> -1) ) THEN SET chave = Id;
    END IF;

    RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK do destino de alta
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKreason;
```

```
DELIMITER $$
CREATE FUNCTION getFKreason (Id INT,Reason VARCHAR(33)) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE chave INT;
    IF (Id NOT IN ( SELECT a1.idReason FROM Dim_Reason a1) AND Reason NOT IN (SELECT a2.Description FROM
Dim_Reason a2)) THEN INSERT INTO Dim_Reason(idReason,Description) VALUES (Id,Reason); END IF;

    IF (Reason IN (SELECT a1.Description FROM Dim_Reason a1) ) THEN SET chave = (SELECT a1.idReason FROM
Dim_Reason a1 WHERE a1.Description = Reason); END IF;
    IF (Id IN (SELECT a1.idReason FROM Dim_Reason a1) AND (Id <> -1) ) THEN SET chave = Id; END IF;

    RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK da info
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKinfo;
DELIMITER $$
CREATE FUNCTION getFKinfo(Cod VARCHAR(45),Diag VARCHAR(250)) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE chave INT;

    IF ( (Cod,Diag) NOT IN ( SELECT a1.Cod_Diagnosis, a1.Description FROM Dim_Info a1 ) ) THEN CALL
InserirNovoDimInfo(Cod,Diag); END IF;
    SET chave = (SELECT a1.idInfo FROM Dim_Info a1 WHERE Cod = a1.Cod_Diagnosis AND Diag =
a1.Description );

    RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK do medicamento
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKdrug;
DELIMITER $$
CREATE FUNCTION getFKdrug (Id INT,Description VARCHAR(227)) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;

     IF ( (Id,Description) NOT IN ( SELECT a1.Cod_Drug, a1.Description FROM Dim_Drug a1 ) ) THEN INSERT
INTO Dim_Drug(Cod_Drug,Description) VALUES (Id,Description); END IF;
     SET chave = (SELECT a1.idDrug FROM Dim_Drug a1 WHERE Id = a1.Cod_Drug AND Description =
a1.Description );

    RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK da INTervenção
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKIntervention;
DELIMITER $$
CREATE FUNCTION getFKINTervention (Id INT, Intervention VARCHAR(155)) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE chave INT;

    IF (Id NOT IN ( SELECT a1.id Intervention FROM Dim_ Intervention a1) AND INTervention NOT IN
( SELECT a2.Description FROM Dim_ Intervention a2)) THEN INSERT INTO Dim_ Intervention (id
Intervention,Description) VALUES (Id, Intervention); END IF;
    IF (Intervention IN ( SELECT a1.Description FROM Dim_Intervention a1) ) THEN SET chave = (SELECT
a1.id Intervention FROM Dim_ Intervention a1 WHERE a1.Description = Intervention); END IF;
    IF (Id IN ( SELECT a1.id Intervention FROM Dim_ Intervention a1) AND (Id <> -1) ) THEN SET chave =
Id; END IF;

    RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK do distrito
```

```
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKdistrito;
DELIMITER $$
CREATE FUNCTION getFKdistrito (Distrito VARCHAR(20)) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE chave INT;
    IF (distrito NOT IN ( SELECT a1.District FROM Dim_District a1 ) ) THEN INSERT INTO Dim_District
(District) VALUES (Distrito); END IF;
    SET chave =  (SELECT a1.idDistrict FROM Dim_District a1 WHERE distrito = a1.District);

    RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK do paciente
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKpatient;
DELIMITER $$
CREATE FUNCTION getFKpatient(Sexo VARCHAR(1), DataNascimento VARCHAR(100), Distrito VARCHAR(20)) RETURNS
INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     INSERT INTO Dim_Patient(Sex,FK_Date_Of_Birth,FK_District) VALUES (Sexo, (SELECT
getFKdata(DataNascimento)), (SELECT getFKdistrito(Distrito)));
     SET chave = last_insert_id();

    RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK do exame
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKexame;
DELIMITER $$
CREATE FUNCTION getFKexame ( Num_Exame VARCHAR(23), Description VARCHAR(104)) RETURNS INT
DETERMINISTIC
BEGIN

 DECLARE chave INT;
 IF (SELECT idUrgency_Exams FROM Dim_Urgency_Exams AS due WHERE due.Num_Exame = Num_Exame AND
due.Description = Description) IS NULL THEN INSERT INTO Dim_Urgency_Exams(Num_Exame, Description) VALUES
(Num_Exame, Description); END IF;
 SET chave = (SELECT idUrgency_Exams FROM Dim_Urgency_Exams AS due WHERE due.Num_Exame = Num_Exame AND
due.Description = Description);

  RETURN chave;

END $$
DELIMITER ;

-- -----------------------------------------------------
-- Função para obter a FK da causa externa
-- -----------------------------------------------------
DROP FUNCTION IF EXISTS getFKextCause;
DELIMITER $$
CREATE FUNCTION getFKextCause (Id INT,Cause VARCHAR(24)) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE chave INT;

    IF (Id NOT IN ( SELECT a1.idExternal_Cause FROM Dim_External_Cause a1) AND Cause NOT IN ( SELECT
a2.Description FROM Dim_External_Cause a2)) THEN INSERT INTO Dim_External_Cause(idExternal_Cause,
Description) VALUES (Id,Cause); END IF;
    IF (Cause IN ( SELECT a1.Description FROM Dim_External_Cause a1) ) THEN SET chave = (SELECT
a1.idExternal_Cause FROM Dim_External_Cause a1 WHERE a1.Description = Cause); END IF;
    IF (Id IN ( SELECT a1.idExternal_Cause FROM Dim_External_Cause a1) AND (Id <> -1) ) THEN SET chave =
Id; END IF;

    RETURN chave;
END $$
DELIMITER ;

-- -----------------------------------------------------
-- TRIGGERS PARA IMPEDIR INFORMAÇÃO REPETIDA
```

```
-- -----------------------------------------------------
DROP TRIGGER IF EXISTS IntegridadeTriageFact;
DELIMITER $
CREATE TRIGGER IntegridadeTriageFact BEFORE INSERT ON Fact_Triage
   FOR EACH ROW
    BEGIN
      IF (new.Urg_Episode in (SELECT a1.Urg_Episode FROM Fact_Triage a1)) THEN SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT='Este Episódio de Urgência já se encontra associado a uma triagem.'; END IF;
    END $
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadeNovaDate;
DELIMITER $
CREATE TRIGGER IntegridadeNovaDate BEFORE INSERT ON Dim_Date
  FOR EACH ROW
    BEGIN
       IF ( new.Date in (SELECT a1.Date FROM Dim_Date a1)) THEN SIGNAL SQLSTATE '45000' SET
MESSAGE_TEXT='Esta data já se encontra na Base de Dados.'; END IF;
   END $
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadeNovaCor;
DELIMITER $
CREATE TRIGGER IntegridadeNovaCor BEFORE INSERT ON Dim_Color
  FOR EACH ROW
    BEGIN
      IF (new.idColor in (SELECT a1.idColor FROM Dim_Color a1)) THEN SIGNAL SQLSTATE '45000' SET
MESSAGE_TEXT='Esta cor já se encontra na Base de Dados'; END IF;
    END $
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadeDiagnosisFact;
DELIMITER $
CREATE TRIGGER IntegridadeDiagnosisFact BEFORE INSERT ON Fact_Diagnosis
  FOR EACH ROW
    BEGIN
       IF (new.Urg_Episode in (SELECT a1.Urg_Episode FROM Fact_Diagnosis a1)) THEN SIGNAL SQLSTATE
'45000' SET MESSAGE_TEXT='Já existe um diagnóstico com este Episódio de Urgência';
        END IF;
    END $
DELIMITER ;

DROP TRIGGER IF EXISTS TwoSexes;
DELIMITER $
CREATE TRIGGER TwoSexes BEFORE INSERT ON Dim_Patient
  FOR EACH ROW
   BEGIN
    IF (new.Sex <> 'F' AND new.Sex <> 'M'AND new.Sex <> 'N')  THEN SIGNAL SQLSTATE '45000' SET
MESSAGE_TEXT='Sexo escolhido não é válido, use F ou M';
  END IF;
END $
DELIMITER ;

DROP TRIGGER IF EXISTS DataValida;
DELIMITER $
CREATE TRIGGER DataValida BEFORE INSERT ON Dim_Patient
FOR EACH ROW
BEGIN
   DECLARE novaData DATETIME; SET novaData = (SELECT a1.Date FROM Dim_Date a1 WHERE new.FK_Date_Of_Birth
= a1.idDate);
   IF (novaData > CURRENT_TIMESTAMP()) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Data Invalida.';
   END IF;
END $
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadeUrgencyEpisodesFact;
DELIMITER $
CREATE TRIGGER IntegridadeUrgencyEpisodesFact BEFORE INSERT ON Fact_Urgency_Episodes
  FOR EACH ROW
    BEGIN
     IF ( new.Urg_Episode in ( SELECT a1.Urg_Episode FROM Fact_Urgency_Episodes a1 ) ) THEN SIGNAL
SQLSTATE '45000' SET MESSAGE_TEXT='Já existe Episódio de Urgência com este identificador.';
        END IF;
    END$
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadeNovaCausa;
DELIMITER $
```

```
CREATE TRIGGER IntegridadeNovaCausa BEFORE INSERT ON Dim_External_Cause
  FOR EACH ROW
  BEGIN
   IF (new.Description in (SELECT a1.Description FROM Dim_External_Cause a1)) THEN SIGNAL SQLSTATE
‘45000’ SET MESSAGE_TEXT=‘Esta causa já se encontra na Base de Dados.’; END IF;
  END $
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadeIntervencao;
DELIMITER $
CREATE TRIGGER IntegridadeIntervenção BEFORE INSERT ON Dim_Intervention
   FOR EACH ROW
    BEGIN
        IF (new.idIntervention in (SELECT a1.idIntervention FROM Dim_Intervention a1) ) THEN SIGNAL
SQLSTATE ‘45000’ SET MESSAGE_TEXT=‘Esta intervenção já se encontra na Base de Dados’; END IF;
    END $
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadeDistricto;
DELIMITER $
CREATE TRIGGER IntegridadeDistricto BEFORE INSERT ON Dim_District
  FOR EACH ROW
   BEGIN
        IF (new.District in (SELECT a1.District FROM Dim_District a1)) THEN SIGNAL SQLSTATE ‘45000’ SET
MESSAGE_TEXT=‘Este distrito já se encontra na Base de Dados.’;END IF;
   END $
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadeDrug;
DELIMITER $
CREATE TRIGGER IntegridadeDrug BEFORE INSERT ON Dim_Drug
  FOR EACH ROW
   BEGIN
        IF (new.Cod_Drug in (SELECT a1.Cod_Drug FROM Dim_Drug a1)) THEN SIGNAL SQLSTATE ‘45000’ SET
MESSAGE_TEXT=‘Este fármaco já se encontra na Base de Dados’; END IF;
   END $
DELIMITER ;

DROP TRIGGER IF EXISTS IntegridadePrescription;
DELIMITER $
CREATE TRIGGER IntegridadePrescription BEFORE INSERT ON Dim_Urgency_Prescription
  FOR EACH ROW
    BEGIN
       IF (new.Cod_Prescription in (SELECT a1.Cod_Prescription FROM Dim_Urgency_Prescription a1)) THEN
SIGNAL SQLSTATE ‘45000’ SET MESSAGE_TEXT= ‘Este código de prescrição já se encontra na Base de Dados’;
       END IF;
  END $
DELIMITER ;
```

Anexo 7 | Atualização do Data Warehouse com os dados presentes na Área de Retenção

```
-- Atualização do Data Warehouse com os dados presentes na area de retenção
use urgency_retencao;

-- Inserção das novas datas no DW
DROP PROCEDURE IF EXISTS InserirDimDate;
DELIMITER $$
CREATE PROCEDURE InserirDimDate()
BEGIN

  DECLARE size INT; DECLARE i INT;
    DECLARE d DATETIME;
    SET size = (SELECT COUNT(*) FROM Dim_Date); SET i = 0;
    DROP VIEW  IF EXISTS vDatas;
    CREATE VIEW vDatas AS SELECT Date FROM `urgency-t2`.Dim_Date;

    WHILE(i<=size) DO
      SET d = (SELECT Date FROM Dim_Date WHERE idDate=i);
      IF(d NOT IN (SELECT * FROM vDatas)) THEN INSERT INTO `urgency-t2`.Dim_Date(Date) SELECT d; END IF;
      SET i = i+1;
    END WHILE;
    DROP VIEW vDatas;
END $$
DELIMITER ;

-- -------------------------------- Triage ------------------------
-- Inserção das novas Cores no DW
DROP PROCEDURE IF EXISTS InserirDim_Color;
DELIMITER $$
CREATE PROCEDURE InserirDim_Color()
BEGIN
    DECLARE size INT;
    DECLARE i INT;
    DECLARE d VARCHAR(8);
    SET size = (SELECT COUNT(*) FROM Dim_Color);
    SET i = 0;
    DROP VIEW  IF EXISTS vColor;
    CREATE VIEW vColor AS SELECT Description FROM `urgency-t2`.Dim_Color;

    WHILE(i<size) DO
        SET d = (SELECT Description d FROM Dim_Color LIMIT i,1);
        IF(d NOT IN (SELECT * FROM vColor)) THEN INSERT INTO `urgency-t2`.Dim_Color SELECT * FROM
Dim_Color WHERE Dim_Color.Description=d; END IF;
        SET i = i+1;
    END WHILE;
    DROP VIEW  vColor;
END $$
DELIMITER ;

-- Inserção de novAS triagens no DW
DROP PROCEDURE IF EXISTS InserirFact_Triage;
DELIMITER $$
CREATE PROCEDURE InserirFact_Triage()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE u INT; DECLARE d DATETIME; DECLARE c VARCHAR(8);
    SET size = (SELECT COUNT(*) FROM Fact_Triage);
    SET i = 0;
    DROP VIEW  IF EXISTS vTriage;
    CREATE VIEW vTriage AS SELECT Urg_Episode FROM `urgency-t2`.Fact_Triage;

    WHILE(i<size) DO
        SET u = (SELECT Urg_Episode FROM Fact_Triage LIMIT i,1);
        SET d = (SELECT Date FROM Dim_Date dD JOIN Fact_Triage fT ON dD.idDate=FT.FK_Date_Admission
WHERE fT.Urg_Episode=u);
        SET c = (SELECT Description FROM Dim_Color dC JOIN Fact_Triage fT ON dC.idColor=FT.FK_Color
WHERE fT.Urg_Episode=u);
        IF(u NOT IN (SELECT * FROM vTriage)) THEN INSERT INTO `urgency-t2`.Fact_Triage VALUES (u,(SELECT
Prof_Triagem FROM Fact_Triage LIMIT i,1),(SELECT Pain_Scale FROM Fact_Triage LIMIT i,1),
        getFKdata2(d),getFKcor2(c)); END IF;
        SET i = i+1;
    END WHILE;
    DROP VIEW  vTriage;
END $$
DELIMITER ;
```

```
-- ---------------------------------------- DIAGNOSIS ----------------------
-- Inserir no DW Dim_Reason
DROP PROCEDURE IF EXISTS InserirDimReason;
DELIMITER $$
CREATE PROCEDURE InserirDimReason()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE r VARCHAR(33);
    SET size = (SELECT COUNT(*) FROM Dim_Reason);
    SET i = 0;
    DROP VIEW  IF EXISTS vReason;
    CREATE VIEW vReason AS SELECT Description FROM `urgency-t2`.Dim_Reason;

    WHILE(i<size) DO
      SET r = (SELECT Description FROM Dim_Reason LIMIT i,1);
        IF(r NOT IN (SELECT * FROM vReason)) THEN INSERT INTO `urgency-t2`.Dim_Reason SELECT * FROM
Dim_Reason WHERE Dim_Reason.Description=r; END IF;
      SET i = i+1;
    END WHILE;
    DROP VIEW vReason;
END $$
DELIMITER ;

-- Inserir no DW Dim_Reason
DROP PROCEDURE IF EXISTS InserirDimDestination;
DELIMITER $$
CREATE PROCEDURE InserirDimDestination()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE d VARCHAR(43);
    SET size = (SELECT COUNT(*) FROM Dim_Destination);
    SET i = 0;
    DROP VIEW  IF EXISTS vReason;
    CREATE VIEW vDestination AS SELECT Description FROM `urgency-t2`.Dim_Destination;

    WHILE(i<size) DO
      SET d = (SELECT Description FROM Dim_Destination LIMIT i,1);
      IF(d NOT IN (SELECT * FROM vDestination)) THEN INSERT INTO `urgency-t2`.Dim_Destination SELECT *
FROM Dim_Destination WHERE Dim_Destination.Description=d; END IF;
        SET i = i+1;
    END WHILE;
    DROP VIEW  vDestination;
END $$
DELIMITER ;

-- Inserção de novos Infos no DW
DROP PROCEDURE IF EXISTS InserirDim_Info;
DELIMITER $$
CREATE PROCEDURE InserirDim_Info()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE cod VARCHAR(45);
    DECLARE idL INT; DECLARE levC VARCHAR(15); DECLARE descrip VARCHAR(250);
    SET size = (SELECT COUNT(*) FROM Dim_Info);
    SET i = 0;
    DROP VIEW  IF EXISTS vInfo;
    CREATE VIEW vInfo AS SELECT Cod_Diagnosis,Description FROM `urgency-t2`.Dim_Info;

    WHILE(i<size) DO
        SET cod = (SELECT Cod_Diagnosis FROM Dim_Info LIMIT i,1);
        SET descrip = (SELECT Description FROM Dim_Info LIMIT i,1);
        SET idL = (SELECT FK_LevelID FROM Dim_Info LIMIT i,1);
        SET levC = (SELECT FK_LevelCod FROM Dim_Info LIMIT i,1);
        IF(cod NOT IN (SELECT Cod_Diagnosis FROM vInfo) AND descrip NOT IN (SELECT Description FROM
vInfo)) THEN INSERT INTO `urgency-t2`.Dim_Info (Cod_Diagnosis,Description,FK_LevelID,FK_LevelCod) VALUES
        (cod,descrip,idL,levC); END IF;
        SET i = i+1;
    END WHILE;
    DROP VIEW  vInfo;
END $$
DELIMITER ;

-- Inserção de novos Diagnosticos no DW
DROP PROCEDURE IF EXISTS InserirFact_Diagnosis;
DELIMITER $$
CREATE PROCEDURE InserirFact_Diagnosis()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE u INT; DECLARE d DATETIME; DECLARE d2 DATETIME;
    DECLARE dest VARCHAR(43); DECLARE reason VARCHAR(33); DECLARE codInfo VARCHAR(45); DECLARE descInfo
VARCHAR(250);
    DECLARE c VARCHAR(8);
```

```
    SET size = (SELECT COUNT(*) FROM Fact_Triage);
    SET i = 0;
    DROP VIEW  IF EXISTS vDiagnosis;
    CREATE VIEW vDiagnosis AS SELECT Urg_Episode FROM `urgency-t2`.Fact_Diagnosis;
WHILE(i<size) DO
  SET u = (SELECT Urg_Episode FROM Fact_Diagnosis LIMIT i,1);
  SET d = (SELECT Date FROM Dim_Date dD JOIN Fact_Diagnosis fD ON dD.idDate=FD.FK_Date_Diagnosis WHERE
fD.Urg_Episode=u);
  SET d2 = (SELECT Date FROM Dim_Date dD JOIN Fact_Diagnosis fD ON dD.idDate=FD.FK_Date_Discharge WHERE
fD.Urg_Episode=u);
  SET dest = (SELECT Description FROM Dim_Destination dD JOIN Fact_Diagnosis fD ON
dD.idDestination=FD.FK_Destination WHERE fD.Urg_Episode=u);
  SET reason = (SELECT Description FROM Dim_Reason dR JOIN Fact_Diagnosis fD ON dR.idReason=FD.FK_Reason
WHERE fD.Urg_Episode=u);
  SET codinfo = (SELECT Cod_Diagnosis FROM Dim_Info dI JOIN Fact_Diagnosis fD ON dI.idInfo=FD.FK_Info
WHERE fD.Urg_Episode=u);
  SET descinfo = (SELECT Description FROM Dim_Info dI JOIN Fact_Diagnosis fD ON dI.idInfo=FD.FK_Info
WHERE fD.Urg_Episode=u);
  IF(u NOT IN (SELECT * FROM vDiagnosis)) THEN INSERT INTO `urgency-t2`.Fact_Diagnosis VALUES (u,(SELECT
Prof_Diagnosis FROM Fact_Diagnosis LIMIT i,1),(SELECT Prof_Discharge FROM Fact_Diagnosis LIMIT i,1),
getFKdata2(d), getFKdestination2(dest), getFKdata2(d2),
getFKreason2(reason),getFKinfo2(codinfo,descinfo)); END IF;
  SET i = i+1;
  END WHILE;
  DROP VIEW  vDiagnosis;
END $$
DELIMITER ;

-- ------------------------------ EPISODES ------------------------------
-- inserir District no DW
DROP PROCEDURE IF EXISTS InserirDimDistrict;
DELIMITER $$
CREATE PROCEDURE InserirDimDistrict()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE d VARCHAR(20);
    SET size = (SELECT COUNT(*) FROM Dim_District);
    SET i = 1;
    DROP VIEW  IF EXISTS vDistrict;
    CREATE VIEW vDistrict AS SELECT District FROM `urgency-t2`.Dim_District;

    WHILE(i<=size) DO
      SET d = (SELECT District FROM Dim_District WHERE idDistrict=i);
      IF(d NOT IN (SELECT * FROM vDistrict)) THEN INSERT INTO `urgency-t2`.Dim_District(District) VALUES
(d); END IF;
      SET i = i+1;
    END WHILE;
    DROP VIEW  vDistrict;
END $$
DELIMITER ;

-- inserir Patient DW
DROP PROCEDURE IF EXISTS inserirDim_Patient;
DELIMITER $$
CREATE PROCEDURE inserirDim_Patient()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE j INT; DECLARE dataN DATETIME; DECLARE dist VARCHAR(20);
    DECLARE sexo VARCHAR(1);
    SET size = (SELECT COUNT(*) FROM Dim_Patient);
    SET i = 1;
    DROP VIEW  IF EXISTS vDistrict;
    CREATE VIEW vDistrict AS SELECT District FROM `urgency-t2`.Dim_District;

    WHILE(i<=size) DO
       SET j = (i-1);
       SET sexo = (SELECT Sex FROM Dim_Patient WHERE idPatient=1);
       SET dataN = (SELECT Date FROM Dim_Date dD JOIN Dim_Patient dP ON dD.idDate=dP.FK_Date_Of_Birth
LIMIT j,1);
       SET dist = (SELECT District FROM Dim_District dD JOIN Dim_Patient dP ON
dD.idDistrict=dP.FK_District LIMIT j,1);
INSERT INTO `urgency-t2`.Dim_Patient(Sex,FK_Date_Of_Birth,FK_District) VALUES
(sexo,getFKdata2(dataN),getFKdistrict2(dist));
SET i = i+1;
END WHILE;
DROP VIEW vDistrict;
END $$
DELIMITER ;

-- inserção de Drugs no DW
```

```
DROP PROCEDURE IF EXISTS InserirDimDrug;
DELIMITER $$
CREATE PROCEDURE InserirDimDrug()
BEGIN
   DECLARE size INT; DECLARE i INT; DECLARE cD INT; DECLARE d VARCHAR(227);
   SET size = (SELECT COUNT(*) FROM Dim_Drug);
   SET i = 0;
   DROP VIEW  IF EXISTS vDrug;
   CREATE VIEW vDrug AS SELECT Cod_Drug,Description FROM `urgency-t2`.Dim_Drug;

    WHILE(i<=size) DO
     SET cD = (SELECT Cod_Drug FROM Dim_Drug WHERE idDrug=i);
     SET d = (SELECT Description FROM Dim_Drug WHERE idDrug=i);
     IF(cD NOT IN (SELECT Cod_Drug FROM vDrug) AND d NOT IN (SELECT Description FROM vDrug)) THEN INSERT
INTO `urgency-t2`.Dim_Drug(Cod_Drug, Description) VALUES (cD,d); END IF;
     SET i = i+1;
END WHILE;
DROP VIEW  vDrug;
END $$
DELIMITER ;

-- Inserção de Exams
DROP PROCEDURE IF EXISTS InserirDim_Urgency_Exams;
DELIMITER $$
CREATE PROCEDURE InserirDim_Urgency_Exams()
BEGIN
  DECLARE size INT; DECLARE i INT; DECLARE nE VARCHAR(23); DECLARE d VARCHAR(104);
  SET size = (SELECT COUNT(*) FROM Dim_Urgency_Exams);
  SET i = 0;
  DROP VIEW  IF EXISTS vUrgency_Exams;
  CREATE VIEW vUrgency_Exams AS SELECT Num_Exame,Description FROM `urgency-t2`.Dim_Urgency_Exams;

  WHILE(i<=size) DO
    SET nE = (SELECT Num_Exame FROM Dim_Urgency_Exams WHERE idUrgency_Exams=i);
    SET d = (SELECT Description FROM Dim_Urgency_Exams WHERE idUrgency_Exams=i);
    IF(nE NOT IN (SELECT Num_Exame FROM vUrgency_Exams) AND d NOT IN (SELECT Description FROM
vUrgency_Exams)) THEN INSERT INTO `urgency-t2`.Dim_Urgency_Exams(Num_Exame, Description) VALUES (nE,d);
END IF;
    SET i = i+1;
  END WHILE;
 DROP VIEW  vUrgency_Exams;
END $$
DELIMITER ;

DROP PROCEDURE IF EXISTS InserirDimExternal_Cause;
DELIMITER $$
CREATE PROCEDURE InserirDimExternal_Cause()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE d INT;
    SET size = (SELECT COUNT(*) FROM Dim_External_Cause);
    SET i = 0;
    DROP VIEW  IF EXISTS vExternalCause;
    CREATE VIEW vExternalCause AS SELECT idExternal_Cause FROM `urgency-t2`.Dim_External_Cause;

    WHILE(i<size) DO
      SET d = (SELECT idExternal_Cause FROM Dim_External_Cause LIMIT i,1);
      IF(d NOT IN (SELECT * FROM vExternalCause)) THEN INSERT INTO `urgency-t2`.Dim_External_Cause
SELECT * FROM Dim_External_Cause WHERE Dim_External_Cause.idExternal_Cause=d; END IF;
      SET i = i+1;
END WHILE;
DROP VIEW  vExternalCause;
END $$
DELIMITER ;

-- Inserir no DW Dim_Intervention
DROP PROCEDURE IF EXISTS InserirDimIntervention;
DELIMITER $$
CREATE PROCEDURE InserirDimIntervention()
BEGIN
  DECLARE size INT; DECLARE i INT; DECLARE d INT;
  SET size = (SELECT COUNT(*) FROM Dim_Intervention);
  SET i = 0;
  DROP VIEW  IF EXISTS vIntervention;
  CREATE VIEW vIntervention AS SELECT idIntervention FROM `urgency-t2`.Dim_Intervention;

  WHILE(i<size) DO
   SET d = (SELECT idIntervention FROM Dim_Intervention LIMIT i,1);
```

```
    IF(d NOT IN (SELECT * FROM vIntervention)) THEN INSERT INTO `urgency-t2`.Dim_Intervention SELECT *
FROM Dim_Intervention WHERE Dim_Intervention.idIntervention=d; END IF;
        SET i = i+1;
  END WHILE;
  DROP VIEW  vIntervention;
END $$
DELIMITER ;

DROP PROCEDURE IF EXISTS InserirDim_Procedures;
DELIMITER $$
CREATE PROCEDURE InserirDim_Procedures()
BEGIN
 DECLARE size INT; DECLARE i INT; DECLARE idP INT; DECLARE pP INT; DECLARE pC INT; DECLARE c INT;
DECLARE d DATETIME;
 DECLARE d2 DATETIME; DECLARE interv INT;
    SET size = (SELECT COUNT(*) FROM Dim_Procedure);
    SET i = 0;
    DROP VIEW  IF EXISTS vProcedure;
    CREATE VIEW vProcedure AS SELECT idPrescription FROM `urgency-t2`.Dim_Procedure;

    WHILE(i<size) DO
        SET idP = (SELECT idPrescription FROM Dim_Procedure LIMIT i,1);
        SET pP = (SELECT Prof_Procedure FROM Dim_Procedure  LIMIT i,1 );
        SET pC= (SELECT Prof_Cancel FROM Dim_Procedure  LIMIT i,1 );
        SET c = (SELECT Canceled FROM Dim_Procedure  LIMIT i,1 );
        SET d = (SELECT Date FROM Dim_Date dD JOIN Dim_Procedure dP ON dD.idDate=dP.FK_Date_Prescription
LIMIT i,1);
        SET d2 = (SELECT Date FROM Dim_Date dD JOIN Dim_Procedure dP ON dD.idDate=dP.FK_Date_BEGIN LIMIT
i,1);
        SET interv  = (SELECT idIntervention FROM Dim_Intervention dI JOIN Dim_Procedure dP ON
dI.idIntervention=dP.FK_Intervention LIMIT i,1);
        IF(idP NOT IN (SELECT * FROM vProcedure)) THEN INSERT INTO `urgency-t2`.Dim_Procedures VALUES
(idP,pP,pC,c,getFKdata2(d),getFKdata2(d2),idIntervention); END IF;
        SET i = i+1;
    END WHILE;
    DROP VIEW  vProcedure;
END $$
DELIMITER ;

-- Inserção de ugency prescriptions no DW
DROP PROCEDURE IF EXISTS InserirUrency_Prescriptions;
DELIMITER $$
CREATE PROCEDURE InserirUrency_Prescriptions()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE cP INT; DECLARE pP INT; DECLARE d DATETIME;
    SET size = (SELECT COUNT(*) FROM Dim_Urgency_Prescription);
    SET i = 0;
    DROP VIEW  IF EXISTS vUP;
    CREATE VIEW vUP AS SELECT Cod_Prescription,Prof_Prescription FROM `urgency-
t2`.Dim_Urgency_Prescription;

    WHILE(i<=size) DO
       SET cP = (SELECT Cod_Prescription FROM Dim_Urgency_Prescription WHERE idUrgency_Prescription=i);
       SET pP= (SELECT Prof_Prescription FROM Dim_Urgency_Prescription WHERE idUrgency_Prescription=i);
       SET d = (SELECT Date FROM Dim_Date dD JOIN Dim_Urgency_Prescription duP ON
dD.idDate=duP.FK_Date_Prescription WHERE idUrgency_Prescription=i);
       IF(cP NOT IN (SELECT Cod_Prescription FROM vUP) AND pP NOT IN (SELECT Prof_Prescription FROM
vUP)) THEN INSERT INTO `urgency-
t2`.Dim_Urgency_Prescription(Cod_Prescription,Prof_Prescription,FK_Date_Prescription)
VALUES(cP,pP,getFKdata2(d)); END IF;
       SET i = i+1;
    END WHILE;
    DROP VIEW  vUP;
END $$
DELIMITER ;

-- Inserção de Prescription_Drug no DW -----------
DROP PROCEDURE IF EXISTS inserirPrescription_Drug;
DELIMITER $$
CREATE PROCEDURE inserirPrescription_Drug()
BEGIN
    DECLARE size INT; DECLARE i INT; DECLARE uP INT; DECLARE d INT; DECLARE QT INT;
    SET size = (SELECT COUNT(*) FROM Prescription_Drug);
    SET i = 0;
    DROP VIEW  IF EXISTS vPD;
    CREATE VIEW vPD AS SELECT * FROM `urgency-t2`.Prescription_Drug;
    WHILE(i<=size) DO
```

```
     SET d = (SELECT(getFKPrescrptionDrugDrug((SELECT Cod_Drug FROM Dim_Drug dD JOIN Prescription_Drug
pD ON dD.idDrug=pD.Drug LIMIT i,1),(SELECT Description FROM Dim_Drug dD JOIN Prescription_Drug pD ON
dD.idDrug=pD.Drug LIMIT i,1))));
    SET uP = (SELECT(getFKPrescrptionDrugPrescription((SELECT Cod_Prescription FROM
Dim_Urgency_Prescription dUP JOIN Prescription_Drug pD ON
dUP.idUrgency_Prescription=pD.Urgency_Prescription LIMIT i,1),(SELECT Prof_Prescription FROM
Dim_Urgency_Prescription dUP JOIN Prescription_Drug pD ON
dUP.idUrgency_Prescription=pD.Urgency_Prescription LIMIT i,1))));
    SET QT = (SELECT Quantity FROM Prescription_Drug LIMIT i,1);
    IF((uP,d,QT)NOT IN (SELECT a1.Urgency_Prescription,a1.Drug,a1.Quantity FROM Prescription_Drug a1))
THEN INSERT INTO `urgency-t2`.Prescription_Drug VALUES (uP,d,QT); END IF;
    SET i = i+1;
END WHILE;
DROP VIEW  vPD;
END $$
DELIMITER ;

-- Inserção de novos Diagnosticos no DW
DROP PROCEDURE IF EXISTS InserirFact_Episodes;
DELIMITER $$
CREATE PROCEDURE InserirFact_Episodes()
BEGIN
DECLARE size INT; DECLARE i INT; DECLARE u INT; DECLARE d DATETIME; DECLARE profA INT; DECLARE pac INT;
DECLARE eC VARCHAR(24); DECLARE uENum VARCHAR(23); DECLARE uED VARCHAR(104); DECLARE c VARCHAR(8);
DECLARE pacDate DATETIME; DECLARE pacDist VARCHAR(20); DECLARE pacSex VARCHAR(1); DECLARE pId INT;
DECLARE uPCod INT;
SET size = (SELECT COUNT(*) FROM Fact_Triage);
SET i = 0;
DROP VIEW  IF EXISTS vFactEpisodes;
CREATE VIEW vFactEpisodes AS SELECT Urg_Episode FROM `urgency-t2`.Fact_Urgency_Episodes;

    WHILE(i<size) DO
        SET u = (SELECT Urg_Episode FROM Fact_Urgency_Episodes LIMIT i,1);
        SET profA = (SELECT Prof_Admission FROM Fact_Urgency_Episodes LIMIT i,1);
        SET pacSex = (SELECT Sex FROM Dim_Patient dP JOIN Fact_Urgency_Episodes fUE ON
dP.idPatient=fUE.FK_Patient WHERE fUE.Urg_Episode=u);
        SET pacDate = (SELECT Date FROM Dim_Patient dP JOIN Fact_Urgency_Episodes fUE ON
dP.idPatient=fUE.FK_Patient JOIN Dim_Date dD ON dD.idDate=dP.FK_Date_Of_Birth WHERE fUE.Urg_Episode=u);
        SET pacDist = (SELECT District FROM Dim_Patient dP JOIN Fact_Urgency_Episodes fUE ON
dP.idPatient=fUE.FK_Patient JOIN Dim_District dD ON dD.idDistrict=dP.FK_District WHERE
fUE.Urg_Episode=u);
        SET d = (SELECT Date FROM Dim_Date dD JOIN Fact_Urgency_Episodes fUE ON
dD.idDate=fUE.FK_Date_Admission WHERE fUE.Urg_Episode=u);
        SET eC = (SELECT Description FROM Dim_External_Cause dE JOIN Fact_Urgency_Episodes fUE ON
dE.idExternal_Cause=fUE.FK_External_Cause WHERE fUE.Urg_Episode=u);
        SET uENum = (SELECT Num_Exame FROM Dim_Urgency_Exams dUE JOIN Fact_Urgency_Episodes fUE ON
dUE.idUrgency_Exams=fUE.FK_Urgency_Exams WHERE fUE.Urg_Episode=u);
        SET uED = (SELECT Description FROM Dim_Urgency_Exams dUE JOIN Fact_Urgency_Episodes fUE ON
dUE.idUrgency_Exams=fUE.FK_Urgency_Exams WHERE fUE.Urg_Episode=u);
        SET pId = (SELECT idPrescription FROM Dim_Procedure dP JOIN Fact_Urgency_Episodes fUE ON
dP.idPrescription=fUE.FK_Procedure WHERE fUE.Urg_Episode=u);
        SET uPCod = (SELECT Cod_Prescription FROM Dim_Urgency_Prescription dUP JOIN
Fact_Urgency_Episodes fUE ON dUP.idUrgency_Prescription=fUE.FK_Urgency_Prescription WHERE
fUE.Urg_Episode=u);

        IF(u NOT IN (SELECT * FROM vFactEpisodes)) THEN INSERT INTO `urgency-t2`.Fact_Urgency_Episodes
VALUES (u,profA,getFKPatient(pacSex,pacDate,pacDist),getFKdata2(d), getFKExternalCause(eC),
getFKUrgency_Exams2(uENum,uED), pID,getFKUrgency_Prescription2(uPCod)); END IF;
        SET i = i+1;
    END WHILE;
    DROP VIEW  vFactEpisodes;
END $$
DELIMITER ;

-- Função para obter a primary key da data
DROP FUNCTION IF EXISTS getFKdata2;
DELIMITER $$
CREATE FUNCTION getFKdata2 ( DataNascimento DATETIME ) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     IF ( DataNascimento NOT IN ( SELECT a1.Date FROM `urgency-t2`.Dim_Date a1) ) THEN INSERT INTO
`urgency-t2`.Dim_Date (date) VALUES (DataNascimento);
    END IF;
    SET chave =  (SELECT a1.idDate FROM `urgency-t2`.Dim_Date a1 WHERE DataNascimento = a1.Date);

   RETURN chave;
END $$
```

```
DELIMITER ;

DROP FUNCTION IF EXISTS getFKcor2;
DELIMITER $$
CREATE FUNCTION getFKcor2 (Cor VARCHAR(8) ) RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE chave INT;
   SET chave = (SELECT idColor FROM `urgency-t2`.Dim_Color d WHERE d.Description=Cor );

   RETURN chave;
END $$
DELIMITER ;

DROP FUNCTION IF EXISTS getFKdestination2;
DELIMITER $$
CREATE FUNCTION getFKdestination2 (dest VARCHAR(43) ) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idDestination FROM `urgency-t2`.Dim_Destination d WHERE d.Description=dest );

     RETURN chave;
END $$
DELIMITER ;
DROP FUNCTION IF EXISTS getFKreason2;
DELIMITER $$
CREATE FUNCTION getFKreason2 (reason VARCHAR(33) ) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idReason FROM `urgency-t2`.Dim_Reason d WHERE d.Description=reason );

    RETURN chave;
END $$
DELIMITER ;

DROP FUNCTION IF EXISTS getFKinfo2;
DELIMITER $$
CREATE FUNCTION getFKinfo2 (codinfo VARCHAR(45),descinfo VARCHAR(250)) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idInfo FROM `urgency-t2`.Dim_Info d WHERE d.Cod_Diagnosis=codInfo AND
d.Description=descInfo);

     RETURN chave;
END $$
DELIMITER ;

DROP FUNCTION IF EXISTS getFKdistrict2;
DELIMITER $$
CREATE FUNCTION getFKdistrict2 (dist VARCHAR(20)) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idDim_District FROM `urgency-t2`.Dim_District d WHERE d.District=dist);

     RETURN chave;
END $$
DELIMITER ;


DROP FUNCTION IF EXISTS getFKPrescrptionDrugDrug;
DELIMITER $$
CREATE FUNCTION getFKPrescrptionDrugDrug (cod INT,descrip VARCHAR(227)) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idDrug FROM `urgency-t2`.Dim_Drug d WHERE d.Cod_Drug=cod AND
d.Description=descrip);

     RETURN chave;
END $$
DELIMITER ;

DROP FUNCTION IF EXISTS getFKPrescrptionDrugPrescription;
DELIMITER $$
```

```
CREATE FUNCTION getFKPrescrptionDrugPrescription (cod INT,prof INT) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idUrgency_Prescription FROM `urgency-t2`.Dim_Urgency_Prescription d WHERE
d.Cod_Prescription=cod AND d.Prof_Prescription=prof);

     RETURN chave;
END $$
DELIMITER ;

DROP FUNCTION IF EXISTS getFKPatient;
DELIMITER $$
CREATE FUNCTION getFKPatient (sex VARCHAR(1),data DATETIME, dist VARCHAR(20)) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idPatient FROM `urgency-t2`.Dim_Patient d WHERE Sex=sex AND
getFKdata2(data)=FK_Date_Of_Birth AND getFKdistrict2(dist)=FK_District LIMIT 1);

    RETURN chave;
END $$
DELIMITER ;

DROP FUNCTION IF EXISTS getFKExternalCause;
DELIMITER $$
CREATE FUNCTION getFKExternalCause (descrip VARCHAR(24)) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idExternal_Cause FROM `urgency-t2`.Dim_External_Cause d WHERE
d.Description=descrip);

     RETURN chave;
END $$
DELIMITER ;

DROP FUNCTION IF EXISTS getFKUrgency_Exams2;
DELIMITER $$
CREATE FUNCTION getFKUrgency_Exams2 (num VARCHAR(23),descrip VARCHAR(24)) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE chave INT;
    SET chave = (SELECT idUrgency_Exams FROM `urgency-t2`.Dim_Urgency_Exams d WHERE d.Num_Exame=num AND
d.Description=descrip);

   RETURN chave;
END $$
DELIMITER ;

DROP FUNCTION IF EXISTS getFKUrgency_Prescription2;
DELIMITER $$
CREATE FUNCTION getFKUrgency_Prescription2 (cod INT) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE chave INT;
     SET chave = (SELECT idUrgency_Prescription FROM `urgency-t2`.Dim_Urgency_Prescription d WHERE
d.Cod_Prescription=cod);

     RETURN chave;
END $$
DELIMITER ;
```

Anexo 8 | Script de Atualização do Data Warehouse e Limpeza da Área de Retenção

```
use urgency_retencao;

CALL InserirDimDate();

CALL InserirDim_Color();
CALL InserirFact_Triage();

CALL InserirDimReason();
CALL InserirDimDestination();
CALL InserirDim_Info();
CALL InserirFact_Diagnosis();

CALL InserirDimDistrict();
CALL inserirDim_Patient();
CALL InserirDimDrug();
CALL InserirDim_Urgency_Exams();
CALL InserirDimExternal_Cause();
CALL InserirDimIntervention();
CALL InserirDim_Procedures();
CALL InserirUrency_Prescriptions();
CALL inserirPrescription_Drug();
CALL InserirFact_Episodes();

DELETE FROM fact_triage;
DELETE FROM Dim_Color;

DELETE FROM Fact_Diagnosis;
DELETE FROM Dim_Info;
DELETE FROM Dim_Destination;
DELETE FROM Dim_Reason;

DELETE FROM Fact_Urgency_Episodes;
DELETE FROM Prescription_Drug;
DELETE FROM Dim_Urgency_Prescription;
DELETE FROM Dim_Procedure;
DELETE FROM Dim_Dim_Intervention;
DELETE FROM Dim_External_Cause;
DELETE FROM Dim_Urgency_Exams;
DELETE FROM Dim_Drug;
DELETE FROM Dim_Patient;
DELETE FROM Dim_District;
DELETE FROM Dim_Date;
```

Anexo 9 | Migração da Área de Retenção para ficheiros csv

```
USE urgency_retencao;

SELECT DISTINCT a1.Urg_Episode, a2.Num_Exame, a2.Description FROM Fact_Urgency_Episodes a1
INNER JOIN Dim_Urgency_Exams a2 ON a1.FK_Urgency_Exams = a2.idUrgency_Exams
INTO OUTFILE 'exams.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';

SELECT DISTINCT a1.Urg_Episode, a4.Quantity, a2.Prof_Prescription, a3.Date , a5.Description,
a2.Cod_Prescription, a5.Cod_Drug FROM Fact_Urgency_Episodes a1
INNER JOIN Dim_Urgency_Prescription a2 ON a1.FK_Urgency_Prescription = a2.idUrgency_Prescription
INNER JOIN Dim_Date a3 ON a2.FK_Date_Prescription = a3.idDate
INNER JOIN Prescription_Drug a4 ON a2.idUrgency_Prescription = a4.Urgency_Prescription
INNER JOIN Dim_Drug a5 ON a4.Drug = a5.idDrug
INTO OUTFILE 'prescriptions.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';

SELECT DISTINCT a1.Urg_Episode, a2.Prof_Cancel, a2.Prof_Procedure, a2.idPrescription, a3.idIntervention,
a4.Date, a2.Canceled, a5.Date, a3.Description FROM Fact_Urgency_Episodes a1
INNER JOIN Dim_Procedure a2 ON a1.FK_Procedure = a2.idPrescription
INNER JOIN Dim_Intervention a3 ON a2.FK_Intervention = a3.idIntervention
INNER JOIN Dim_Date a4 ON a2.FK_Date_Prescription = a4.idDate
INNER JOIN Dim_Date a5 ON a2.FK_Date_Begin = a5.idDate
INTO OUTFILE 'procedures.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';

SELECT DISTINCT a1.Urg_Episode, a3.Date, a2.Sex, a4.District, a5.Date, a6.idExternal_Cause,
a6.Description, a1.Prof_Admission, t2.Date, t1.Prof_Triagem, t1.Pain_Scale, t3.idColor, t3.Description,
d2.Cod_Diagnosis, d2.Description, d3.Date, d1.Prof_Diagnosis, d4.idDestination, d4.Description,
d1.Prof_Discharge,d5.Date, d6.idReason, d6.Description FROM Fact_Urgency_Episodes a1
INNER JOIN Dim_Patient a2 ON a1.FK_Patient = a2.idPatient
INNER JOIN Dim_Date a3 ON a2.FK_Date_Of_Birth = a3.idDate
INNER JOIN Dim_District a4 ON a2.FK_District = a4.idDistrict
INNER JOIN Dim_Date a5 ON a1.FK_Date_Admission = a5.idDate
INNER JOIN Dim_External_Cause a6 ON a1.FK_External_Cause = a6.idExternal_Cause
INNER JOIN Fact_Triage t1 ON a1.Urg_Episode = t1.Urg_Episode
INNER JOIN Dim_Date t2 ON t1.FK_Date_Admission = t2.idDate
INNER JOIN Dim_Color t3 ON t1.FK_Color = t3.idColor
INNER JOIN Fact_Diagnosis d1 ON a1.Urg_Episode = d1.Urg_Episode
INNER JOIN Dim_Info d2 ON d1.FK_Info = d2.idInfo
INNER JOIN Dim_Date d3 ON d1.FK_Date_Diagnosis = d3.idDate
INNER JOIN Dim_Destination d4 ON d1.FK_Destination = d4.idDestination
INNER JOIN Dim_Date d5 ON d1.FK_Date_Discharge = d5.idDate
INNER JOIN Dim_Reason d6 ON d1.FK_Reason = d6.idReason
INTO OUTFILE 'episodes.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';
```