

# Trabalho Final de SQL para Ciência de Dados - CI245

Análise de Dados com SQL sobre Ocupações dos Leitos de COVID-19 nos Anos de 2020, 2021 e 2022.

## Table of contents

0.1	Introdução . . . . .	2
0.2	Conexão com Banco de Dados . . . . .	2
0.3	Limpeza de Dados . . . . .	2
0.3.1	Criação de views . . . . .	3
0.3.2	Remoção de registros (notificações) com status excluídos . . . . .	3
0.3.3	Remoção (filtro) de registros que continham valores decimais . . . . .	4
0.3.4	Validação de colunas de acordo com informações do dicionário de dados . . . . .	4
0.3.5	Remoção de notificações duplicadas . . . . .	7
0.3.6	Criação de colunas calculadas . . . . .	8
0.3.7	Criação de série temporal . . . . .	9
0.3.8	Execução de ‘Last Observation Carried Forward (LOCF)’ para tratamento de valores nulos em séries temporais. . . . .	10
0.3.9	Criação de <b>materialized view</b> como tabela final para facilitar uso durante consulta dos dados tratados. . . . .	11
0.4	Análise de Dados . . . . .	12
0.4.1	Views e/ou CTE’s . . . . .	12
0.4.2	Subconsultas . . . . .	12
0.4.3	Detecção de Anomalias: registros com picos isolados . . . . .	12
0.4.4	Detecção de Anomalias: registros com picos isolados . . . . .	13
<b>1</b>	<b>Identificação de Dias Atípicos na Quantidade de Óbitos por Covid-19</b>	<b>13</b>
1.0.1	Análise de Texto: origem dos dados . . . . .	16
1.0.2	Evolução das altas e óbitos ao longo do tempo . . . . .	16
1.0.3	Evolução da ocupados COVID vs ocupados hospitalar . . . . .	18
1.0.4	Ocupação máxima de leitos por hospital . . . . .	19
1.1	Observações Finais . . . . .	19

## 0.1 Introdução

Este relatório tem como objetivo realizar a análise de dados sobre a ocupação dos leitos hospitalares destinados ao tratamento de COVID-19 nos anos de 2020, 2021 e 2022, utilizando SQL como ferramenta principal. A análise foi realizada com base em um banco de dados com informações obtidas através de arquivos .csv e estão disponíveis em: <https://opendatasus.saude.gov.br/dataset/registro-de-ocupacao-hospitalar-covid-19>. Tais dizem respeito a ocupação dos leitos, registros de óbitos e outras métricas relevantes.

As principais análises realizadas foram: - Detecção de anomalias: registros com picos isolados; - Análise de Texto: origem dos dados; - Evolução das altas e óbitos ao longo do tempo.

Para tanto foram utilizadas técnicas de **SQL** para manipulação e análise dos dados, com foco na extração de informações relevantes e na identificação de padrões e anomalias.

## 0.2 Conexão com Banco de Dados

```
library(DBI)
library(RPostgres)

# Criando conexão (exemplo PostgreSQL)
con <- dbConnect(
  RPostgres::Postgres(),
  dbname = "ocupacao",
  host = "localhost",
  port = 5432,
  user = "postgres",
  password = senha)
```

## 0.3 Limpeza de Dados

Antes de fazermos qualquer análise estatística, precisamos garantir que os dados estão modelados de tal forma que sejam íntegros e de qualidade. Não à toa, já é clichê dizer que boa parte do trabalho de qualquer analista ou cientista de dados é a limpeza dos dados, afinal a máxima “garbage in, garbage out” é uma realidade.

Essa base de dados, por serem preenchidas por humanos em alguns casos, possui alguns problemas que, **a priori**, dificultaram o nosso entendimento sobre as informações que os dados nos trazem. Lendo o dicionário de dados com acuracidade, conseguimos identificar quais padrões poderiam nos guiar na validação destes dados.

Assim, para o tratamento de dados, foram realizadas algumas etapas. Para cada uma, views foram criadas para ajudar na organização do código. 1 - Remoção (filtro) de registros (notificações) com status excluídos 2 - Remoção (filtro) de registros que continham valores decimais. 3 - Validação de colunas de acordo com informações do dicionário de dados. 4 - Remoção de notificações duplicadas 5 - Criação de colunas calculadas 6 - Criação de série temporal 7 - Execução de ‘Last Observation Carried Forward (LOCF)’ para tratamento de valores nulos em séries temporais. 8 - Criação de **materialized view** como tabela final para facilitar uso durante consulta dos dados tratados.

### 0.3.1 Criação de views

Antes de executarmos o **script** que cria a **materialized view** final, garantimos que já não existe no banco de dados outros objetos com o mesmo nome. Para isso, executamos o comando “**DROP VIEW IF EXISTS nome\_da\_view**”; para cada view que vamos criar.

```
DROP MATERIALIZED VIEW IF EXISTS registro_ocupacao_final;
```

```
DROP VIEW IF EXISTS registro_ocupacao_07_locf;
```

```
DROP VIEW IF EXISTS registro_ocupacao_06_serie_temporal;
```

```
DROP VIEW IF EXISTS registro_ocupacao_05_colunas_calculadas_filtradas;
```

```
DROP VIEW IF EXISTS registro_ocupacao_04_sem_duplicatas;
```

```
DROP VIEW IF EXISTS registro_ocupacao_03_dados_limpos;
```

```
DROP VIEW IF EXISTS registro_ocupacao_02_sem_decimais;
```

```
DROP VIEW IF EXISTS registro_ocupacao_01_nao_excluido;
```

### 0.3.2 Remoção de registros (notificações) com status excluídos

Observamos que alguns registros continham status de exclusão com valores lógicos **true**, o que não faz sentido para a análise. Assim, criamos uma view que filtra esses registros para manter apenas aqueles com status não excluídos.

```
-- Remove registros com status excluidos
CREATE OR REPLACE VIEW registro_ocupacao_01_nao_excluido AS
  SELECT ro.*
  FROM registro_ocupacao ro
  LEFT JOIN status_envio se
    ON ro.id_status = se.id_status
  WHERE excluido = false;
```

### 0.3.3 Remoção (filtro) de registros que continham valores decimais

Alguns valores de ocupação e saída continham valores decimais (valores relativos pois não passavam de 100), o que não faz sentido para a análise, pois esses valores deveriam ser inteiros. Assim, criamos uma view que filtra esses registros para manter apenas aqueles com valores inteiros.

```
-- Remove valores decimais
CREATE OR REPLACE VIEW registro_ocupacao_02_sem_decimais AS
  SELECT * FROM registro_ocupacao_01_nao_excluido
  EXCEPT
  SELECT * FROM registro_ocupacao_01_nao_excluido
  WHERE ocupacao_covid_cli % 1 != 0
    OR ocupacao_covid_uti % 1 != 0
    OR ocupacao_hospitalar_cli % 1 != 0
    OR ocupacao_hospitalar_uti % 1 != 0
    OR saida_suspeita_obitos % 1 != 0
    OR saida_suspeita_altas % 1 != 0
    OR saida_confirmada_obitos % 1 != 0
    OR saida_confirmada_altas % 1 != 0
    OR ocupacao_suspeito_cli % 1 != 0
    OR ocupacao_confirmado_cli % 1 != 0
    OR ocupacao_suspeito_uti % 1 != 0
    OR ocupacao_confirmado_uti % 1 != 0;
```

### 0.3.4 Validação de colunas de acordo com informações do dicionário de dados

De acordo com o dicionário de dados, estabelecemos alguns critérios para validar os valores das colunas de ocupação e saídas:

- Se *ocupacao\_hospitalar\_cli* for igual a zero ou nulo, então, necessariamente, *ocupacao\_covid\_cli* também deve ser igual a zero ou nulo.

```

SELECT
    COUNT(*)::DECIMAL AS sem_logica_count,
    (SELECT COUNT(*)::DECIMAL
     FROM registro_ocupacao_02_sem_decimais) AS total_count,
    ROUND(COUNT(*)::DECIMAL / (SELECT COUNT(*)::DECIMAL
                               FROM registro_ocupacao_02_sem_decimais)
          , 4) AS sem_logica_percentual
FROM registro_ocupacao_02_sem_decimais
WHERE (ocupacao_hospitalar_cli = 0
      AND ocupacao_covid_cli > 0)
OR (ocupacao_hospitalar_cli IS NULL
    AND ocupacao_covid_cli IS NOT NULL);

```

- Se *ocupacao\_hospitalar\_uti* for igual a zero ou nulo, então, necessariamente, *ocupacao\_covid\_uti* também deve ser igual a zero ou nulo.

```

-- Validacao coluna ocupacao_covid_uti
SELECT
    COUNT(*)::DECIMAL AS sem_logica_count,
    (SELECT COUNT(*)::DECIMAL
     FROM registro_ocupacao_02_sem_decimais) AS total_count,
    ROUND(COUNT(*)::DECIMAL / (SELECT COUNT(*)::DECIMAL
                               FROM registro_ocupacao_02_sem_decimais)
          , 4) AS sem_logica_percentual
FROM registro_ocupacao_02_sem_decimais
WHERE (ocupacao_hospitalar_uti = 0
      AND ocupacao_covid_uti > 0)
OR (ocupacao_hospitalar_uti IS NULL
    AND ocupacao_covid_uti IS NOT NULL);

```

- – Se *ocupacao\_covid\_cli* for igual a zero ou nulo, então, necessariamente, *ocupacao\_suspeito\_cli* e *ocupacao\_confirmado\_cli* também devem ser iguais a zero ou nulos.

```

-- Validacao colunas ocupacao_suspeito_cli e ocupacao_confirmado_cli
SELECT
    COUNT(*)::DECIMAL AS sem_logica_count,
    (SELECT COUNT(*)::DECIMAL
     FROM registro_ocupacao_02_sem_decimais) AS total_count,
    ROUND(COUNT(*)::DECIMAL / (SELECT COUNT(*)::DECIMAL
                               FROM registro_ocupacao_02_sem_decimais)
          , 4) AS sem_logica_percentual

```

```

FROM registro_ocupacao_02_sem_decimais
WHERE (ocupacao_covid_cli = 0
      AND (ocupacao_suspeito_cli > 0
           OR ocupacao_confirmado_cli > 0))
OR (ocupacao_covid_cli IS NULL
    AND (ocupacao_suspeito_cli IS NOT NULL
         OR ocupacao_confirmado_cli IS NOT NULL));

```

- – Se *ocupacao\_covid\_cli* for igual a zero ou nulo, então, necessariamente, *ocupacao\_suspeito\_cli* e *ocupacao\_confirmado\_cli* também devem ser iguais a zero ou nulos.

```

-- Validacao colunas ocupacao_suspeito_cli e ocupacao_confirmado_cli
SELECT
COUNT(*)::DECIMAL AS sem_logica_count,
(SELECT COUNT(*)::DECIMAL
 FROM registro_ocupacao_02_sem_decimais) AS total_count,
ROUND(COUNT(*)::DECIMAL / (SELECT COUNT(*)::DECIMAL
                           FROM registro_ocupacao_02_sem_decimais)
      , 4) AS sem_logica_percentual
FROM registro_ocupacao_02_sem_decimais
WHERE (ocupacao_covid_cli = 0
      AND (ocupacao_suspeito_cli > 0
           OR ocupacao_confirmado_cli > 0))
OR (ocupacao_covid_cli IS NULL
    AND (ocupacao_suspeito_cli IS NOT NULL
         OR ocupacao_confirmado_cli IS NOT NULL));

```

Com base nessas validações, percebemos que cerca de 62% dos registros das colunas *ocupacao\_suspeito\_cli*, *ocupacao\_confirmado\_cli*, *ocupacao\_suspeito\_cli* e *ocupacao\_confirmado\_cli* apresentam inconsistências e, por isso, decidimos não utilizá-las nas análises.

Ademais, como menos de 1% dos registros das colunas *ocupacao\_covid\_cli* e *ocupacao\_covid\_cli* apresentam inconsistências, criaremos uma visualização que filtre os valores consistentes dessas colunas, conforme o **script abaixo**.

```

-- valida regras de acordo com interpretacao do dicionario de dados
CREATE OR REPLACE VIEW registro_ocupacao_03_dados_limpos AS
SELECT
    _id, data_notificacao, criado_em, atualizado_em,
    id_hospital, id_local, id_status,

```

```

        ocupacao_covid_cli, ocupacao_covid_uti,
        ocupacao_hospitalar_cli, ocupacao_hospitalar_uti,
        saida_suspeita_obitos, saida_suspeita_altas,
        saida_confirmada_obitos, saida_confirmada_altas
FROM registro_ocupacao_02_sem_decimais
EXCEPT
SELECT
    _id, data_notificacao, criado_em, atualizado_em,
    id_hospital, id_local, id_status,
    ocupacao_covid_cli, ocupacao_covid_uti,
    ocupacao_hospitalar_cli, ocupacao_hospitalar_uti,
    saida_suspeita_obitos, saida_suspeita_altas,
    saida_confirmada_obitos, saida_confirmada_altas
FROM registro_ocupacao_02_sem_decimais
WHERE (ocupacao_hospitalar_cli = 0
        AND ocupacao_covid_cli > 0)
    OR (ocupacao_hospitalar_cli IS NULL
        AND ocupacao_covid_cli IS NOT NULL)
    OR (ocupacao_hospitalar_uti = 0
        AND ocupacao_covid_uti > 0)
    OR (ocupacao_hospitalar_uti IS NULL
        AND ocupacao_covid_uti IS NOT NULL);

```

### 0.3.5 Remoção de notificações duplicadas

Alguns registros continham notificações duplicadas, ou seja, com a mesma data de notificação e o mesmo hospital. Para resolver isso, criamos uma view que remove as duplicatas, mantendo apenas a notificação mais recente (com base na data de atualização e criação).

```

-- Remove notificacoes duplicadas, trazendo aquelas com data de atualizacao mais recente
CREATE OR REPLACE VIEW registro_ocupacao_04_sem_duplicatas AS
WITH ranked AS (
    SELECT *,
        ROW_NUMBER() OVER (
            PARTITION BY data_notificacao, id_hospital
            ORDER BY atualizado_em DESC, criado_em DESC, _id DESC
        ) AS rn
    FROM registro_ocupacao_03_dados_limpos
)
SELECT
    _id, id_hospital, data_notificacao, id_local, id_status,

```

```

ocupacao_covid_cli, ocupacao_covid_uti,
ocupacao_hospitalar_cli, ocupacao_hospitalar_uti,
saida_suspeita_obitos, saida_suspeita_altas,
saida_confirmada_obitos, saida_confirmada_altas
FROM ranked
WHERE rn = 1;

```

### 0.3.6 Criação de colunas calculadas

Para facilitar a análise, criamos colunas calculadas que agregam informações relevantes. As colunas criadas foram: *ocupacao\_covid*, *ocupacao\_hospitalar*, *saida\_covid\_obitos*, *saida\_covid\_altas*, *saida\_covid*, *entrada\_covid\_calculada*.

```

-- Cria colunas calculadas saida_obitos, saida_altas, saida_covid, saida_covid_calculada e e
CREATE OR REPLACE VIEW registro_ocupacao_05_colunas_calculadas_filtradas AS
  WITH com_colunas_calculadas AS(
    SELECT
      _id, id_hospital, data_notificacao, id_local, id_status,

      (ocupacao_covid_cli + ocupacao_covid_uti) AS ocupacao_covid,

      (ocupacao_hospitalar_cli +
      ocupacao_hospitalar_uti) AS ocupacao_hospitalar,

      (COALESCE(saida_suspeita_obitos, 0) +
      COALESCE(saida_confirmada_obitos, 0)) AS saida_covid_obitos,

      (COALESCE(saida_suspeita_altas, 0) +
      COALESCE(saida_confirmada_altas, 0)) AS saida_covid_altas,

      (COALESCE(saida_suspeita_obitos, 0) +
      COALESCE(saida_confirmada_obitos, 0) +
      COALESCE(saida_suspeita_altas, 0) +
      COALESCE(saida_confirmada_altas, 0)) AS saida_covid

    FROM registro_ocupacao_04_sem_duplicatas),

  com_colunas_calculadas2 AS(
    SELECT *,

      ( ocupacao_covid -

```



```

        COALESCE((LAG(ocupacao_covid) OVER(
            ORDER BY id_hospital, data_notificacao)), 0) +
        saida_covid ) AS entrada_covid_calculada

FROM com_colunas_calculadas)

SELECT
    _id, id_hospital,
    data_notificacao, id_local, id_status,
    ocupacao_covid, ocupacao_hospitalar,
    saida_covid_obitos, saida_covid_altas,
    CASE WHEN entrada_covid_calculada < 0
        THEN saida_covid + (-1) * entrada_covid_calculada
        ELSE saida_covid
    END AS saida_covid_calculada,
    CASE
        WHEN entrada_covid_calculada < 0 THEN 0
        ELSE entrada_covid_calculada
    END AS entrada_covid_calculada
FROM com_colunas_calculadas2;

```

### 0.3.7 Criação de série temporal

Para facilitar a análise temporal dos dados, criamos uma série temporal que combina todas as datas de notificação com os hospitais, locais e status. Isso nos permite analisar a evolução dos dados ao longo do tempo, mesmo quando alguns registros estão ausentes.

```

-- Cria série temporal
CREATE OR REPLACE VIEW registro_ocupacao_06_serie_temporal AS
    WITH calendario AS(

        SELECT GENERATE_SERIES(
            MIN(data_notificacao),
            MAX(data_notificacao),
            '1 day') AS data_notificacao

        FROM registro_ocupacao_05_colunas_calculadas_filtradas),

    combinacoes_hospital_local_status AS(
        SELECT DISTINCT id_hospital, id_local, id_status
        FROM registro_ocupacao_05_colunas_calculadas_filtradas
    )

```

```

),

serie_temporal AS (
    SELECT
        cal.data_notificacao AS data_notificacao,
        hls.id_hospital,
        hls.id_local,
        hls.id_status
    FROM calendario cal
    CROSS JOIN combinacoes_hospital_local_status hls
    ORDER BY 2, 3, 4, 1
)

SELECT
    st.data_notificacao AS data_notificacao,
    st.id_hospital,
    st.id_local,
    st.id_status,
    ro.ocupacao_covid,
    ro.ocupacao_hospitalar,
    ro.saida_covid_obitos,
    ro.saida_covid_altas,
    ro.saida_covid_calculada,
    ro.entrada_covid_calculada
FROM serie_temporal st
LEFT JOIN registro_ocupacao_05_colunas_calculadas_filtradas ro
    ON st.id_hospital = ro.id_hospital
    AND st.id_local = ro.id_local
    AND st.id_status = ro.id_status
    AND DATE(st.data_notificacao) = DATE(ro.data_notificacao)
ORDER BY st.id_hospital, st.id_local, st.id_status, st.data_notificacao;

```

### 0.3.8 Execução de ‘Last Observation Carried Forward (LOCF)’ para tratamento de valores nulos em séries temporais.

Para lidar com valores nulos na série temporal, aplicamos a técnica de “Last Observation Carried Forward (LOCF)”, que preenche os valores nulos com o último valor observado (data imediatamente mais recente). Isso é especialmente útil em séries temporais, onde a continuidade dos dados é importante para a análise.

```

-- Executa 'Last Observation Carried Forward (LOCF)' para tratamento de valores nulos em sé
CREATE OR REPLACE VIEW registro_ocupacao_07_LOCF AS
SELECT
    data_notificacao,
    id_hospital,
    id_local,
    id_status,
    CASE
        WHEN ocupacao_covid IS NULL
        THEN LAG(ocupacao_covid) OVER (
            PARTITION BY id_hospital, id_local, id_status
            ORDER BY data_notificacao)
        ELSE ocupacao_covid
    END AS ocupacao_covid,

    CASE
        WHEN ocupacao_hospitalar IS NULL
        THEN LAG(ocupacao_hospitalar) OVER (
            PARTITION BY id_hospital, id_local, id_status
            ORDER BY data_notificacao)
        ELSE ocupacao_hospitalar
    END AS ocupacao_hospitalar,

    COALESCE(saida_covid_obitos, 0) AS saida_covid_obitos,
    COALESCE(saida_covid_altas, 0) AS saida_covid_altas,
    COALESCE(saida_covid_calculada, 0) AS saida_covid_calculada,
    COALESCE(entrada_covid_calculada, 0) AS entrada_covid_calculada
FROM registro_ocupacao_06_serie_temporal;

```

### 0.3.9 Criação de materialized view como tabela final para facilitar uso durante consulta dos dados tratados.

Criamos uma **materialized view** final que contém todos os dados tratados e prontos para análise. Escolhemos esse objeto porque ele permite consultas mais rápidas e eficientes, especialmente em grandes volumes de dados. Essa tabela final é ordenada por hospital, data de notificação, local e status, facilitando a consulta e análise dos dados.

```

-- Cria tabela final com dados prontos para analise
CREATE MATERIALIZED VIEW registro_ocupacao_final AS
SELECT * FROM registro_ocupacao_07_LOCF
ORDER BY id_hospital, data_notificacao, id_local, id_status;

```

## 0.4 Análise de Dados

Com um tema tão amplo, a análise de dados foi dividida em três partes principais: detecção de anomalias, análise de texto e evolução das altas e dos óbitos ao longo do tempo. Nossa curiosidade em explorar mais dados sobre um período tão catastrófico da história da humanidade nos levou, também, a realizar análises adicionais.

### 0.4.1 Views e/ou CTE's

Foram utilizadas **views** e **CTEs** para organizar as consultas SQL e facilitar a análise dos dados. Essas estruturas foram empregadas tanto nos **scripts** de limpeza de dados quanto nas análises que seguem abaixo.

### 0.4.2 Subconsultas

Foram utilizadas subconsultas para a limpeza dos dados.

### 0.4.3 Detecção de Anomalias: registros com picos isolados

```
-- Detecção de Anomalias temporais
CREATE OR MATERIALIZED VIEW anomalias_temporais_1 AS
WITH estat AS (
    SELECT
        id_hospital,
        avg(ocupacao_covid) AS med_c,
        avg(ocupacao_hospitalar) AS med_h,
        STDDEV_POP(ocupacao_covid) AS sd_c,
        STDDEV_POP(ocupacao_hospitalar) AS sd_h
    FROM registro_ocupacao_final
    GROUP BY id_hospital
)

SELECT * FROM estat;
```

```
-- Usando partition by

CREATE OR MATERIALIZED VIEW anomalias_temporais_2 AS
WITH estat AS (
    SELECT
```

```

        id_hospital,
        data_notificacao,
        ocupacao_covid,
        ocupacao_hospitalar,
        AVG(ocupacao_covid) OVER (PARTITION BY id_hospital) AS med_c,
        AVG(ocupacao_hospitalar) OVER (PARTITION BY id_hospital) AS med_h,
        STDDEV_POP(ocupacao_covid) OVER (PARTITION BY id_hospital) AS sd_c,
        STDDEV_POP(ocupacao_hospitalar) OVER (PARTITION BY id_hospital) AS sd_h
    FROM registro_ocupacao_final
)

SELECT
    id_hospital,
    data_notificacao,
    ocupacao_covid,
    ocupacao_hospitalar,
    med_c,
    med_h,
    sd_c,
    sd_h
FROM estat
WHERE ocupacao_covid > 3*sd_c OR ocupacao_hospitalar > 3*sd_h;

```

#### 0.4.4 Detecção de Anomalias: registros com picos isolados

## 1 Identificação de Dias Atípicos na Quantidade de Óbitos por Covid-19

A consulta abaixo busca identificar quais foram os dias mais atípicos em relação à quantidade de óbitos por Covid-19 durante o período da pandemia. Para isso, utilizamos uma técnica estatística que compara o número de óbitos de cada dia com a **média móvel dos últimos  $n$  dias**, tendo como parâmetro o **z-score**.

Para concretizar essa classificação dos dias como anômalos ou não, desenvolvemos um script com os seguintes passos:

1. **Agregação dos óbitos por data**, por meio da CTE `obitos_por_data`.
2. **Cálculo da média móvel e do desvio padrão móvel dos últimos 30 dias**, utilizando a CTE `base`.

3. **Determinação dos limites inferior e superior** que definem se um dia é considerado anômalo, com base no seguinte critério:  
Se a quantidade de óbitos em um determinado dia estiver **acima ou abaixo de três vezes o desvio padrão móvel em relação à média móvel**, então esse dia é classificado como anômalo, sendo:
- 1 para uma anomalia positiva (valores muito acima da média)
  - -1 para uma anomalia negativa (valores muito abaixo da média)
  - 0 para um dia não anômalo (dentro dos limites)
4. Por fim, a consulta retorna todos os dias, juntamente com as métricas calculadas e a indicação de anomalia, permitindo uma análise clara dos períodos mais atípicos no número de óbitos.

```
-- Detecção de Anomalias Temporais

CREATE OR MATERIALIZED VIEW anomalias_temporais_3 AS
WITH obitos_por_data AS(
    SELECT
        DATE_TRUNC('day', data_notificacao) AS data_notificacao,
        SUM(saida_covid_obitos) AS obitos
    FROM registro_ocupacao_final
    GROUP BY 1
    ORDER BY 1
),
base AS (
    SELECT
        data_notificacao,
        obitos,

        -- Média móvel dos últimos 30 dias
        AVG(obitos) OVER (
            ORDER BY data_notificacao
            ROWS BETWEEN 29 PRECEDING AND CURRENT ROW
        ) AS media_movel_obitos,

        -- Desvio padrão móvel dos últimos 30 dias
        STDDEV_POP(obitos) OVER (
            ORDER BY data_notificacao
            ROWS BETWEEN 29 PRECEDING AND CURRENT ROW
        ) AS desvio_movel_obitos
```

```

    FROM obitos_por_data
),

anomalias AS(
    SELECT
        data_notificacao,
        obitos,
        media_movel_obitos,
        desvio_movel_obitos,

        -- Limites dinâmicos
        media_movel_obitos + 3 * desvio_movel_obitos AS limite_superior,
        media_movel_obitos - 3 * desvio_movel_obitos AS limite_inferior,

        -- Z-score
        (obitos - media_movel_obitos) / NULLIF(desvio_movel_obitos, 0) AS z_score,

        -- Verificação de anomalias (picos (1), normalidades (0) ou quedas (-1))
        CASE
            WHEN obitos > media_movel_obitos + 3 * desvio_movel_obitos THEN 1
            WHEN obitos < media_movel_obitos - 3 * desvio_movel_obitos THEN -1
            ELSE 0
        END AS anomalia

    FROM base
    ORDER BY data_notificacao
)
SELECT
    data_notificacao,
    obitos,
    ROUND(media_movel_obitos, 2) AS media_movel_obitos,
    ROUND(desvio_movel_obitos, 2) AS desvio_movel_obitos,
    ROUND(limite_inferior, 2) AS limite_inferior,
    ROUND(limite_superior, 2) AS limite_superior,
    ROUND(z_score, 2) AS z_score,
    anomalia
FROM anomalias
-- WHERE anomalia = 1 OR anomalia = -1
ORDER BY data_notificacao;

```

### 1.0.1 Análise de Texto: origem dos dados

Neste trecho de código, para sabermos quantas observações são provenientes de cada origem dos dados, selecionamos a coluna do id da origem e contamos quantas vezes ele aparece, para então agregar esta informação a outra tabela que contém a coluna de origem dos dados, e por fim, agrupamos todos os valores pela origem dos dados, verificando o volume de informação proveniente de cada origem.

```
CREATE OR MATERIALIZED VIEW analise_textual_origem_dos_dados AS
WITH n_reg AS (
  SELECT
    id_status,
    COUNT(id_status) AS qtde
  FROM registro_ocupacao
  GROUP BY id_status
),
jun AS (
  SELECT
    n_reg.id_status,
    n_reg.qtde,
    status_envio.origem
  FROM n_reg
  JOIN status_envio
    ON n_reg.id_status = status_envio.id_status
),
agrupado AS (
  SELECT
    origem,
    SUM(qtde) AS qtde
  FROM jun
  GROUP BY origem
  ORDER BY qtde DESC
)

SELECT * FROM agrupado;
```

### 1.0.2 Evolução das altas e óbitos ao longo do tempo

# Análise da Evolução Temporal das Altas e Óbitos por Covid-19

Para analisar a evolução temporal das altas e compará-las com a quantidade de óbitos, elaboramos a consulta que segue abaixo.



Ela é composta basicamente por três partes:

1. **\*\*CTE 'truncado'\*\***, onde criamos quatro novas colunas ('dia', 'mes', 'trimestre' e 'ano') a partir do truncamento da data;
2. **\*\*CTEs 'axo\_dia', 'axo\_mes', 'axo\_trimestre' e 'axo\_ano'\*\***, que executam a mesma função: agregam os valores de altas e óbitos conforme o tipo de data truncada correspondente;
3. **\*\*Consulta final\*\***, que seleciona todas as colunas, permitindo ao usuário escolher qual CTE deseja visualizar, de acordo com a granularidade temporal desejada.

Dessa forma, conseguimos observar a evolução temporal, em diferentes granularidades, da quantidade total de óbitos versus altas durante o período da pandemia de Covid-19.

```
-- Evolucao das altas vs. obitos
CREATE OR MATERIALIZED VIEW evolucao_obitos_altas AS
WITH truncado AS (
    SELECT
        DATE(DATE_TRUNC('day', data_notificacao)) AS dia,
        DATE(DATE_TRUNC('month', data_notificacao)) AS mes,
        DATE(DATE_TRUNC('quarter', data_notificacao)) AS trimestre,
        DATE(DATE_TRUNC('year', data_notificacao)) AS ano,
        saida_covid_obitos AS obitos,
        saida_covid_altas AS altas
    FROM registro_ocupacao_final
),
axo_dia AS (
    SELECT
        dia,
        SUM(obitos) AS obitos,
        SUM(altas) AS altas
    FROM truncado
    GROUP BY dia
    ORDER BY dia DESC
),
axo_mes AS (
    SELECT
        mes,
        SUM(obitos) AS obitos,
        SUM(altas) AS altas
    FROM truncado
    GROUP BY mes
    ORDER BY mes DESC
```

```

),
axo_trimestre AS (
  SELECT
    trimestre,
    SUM(obitos) AS obitos,
    SUM(alta) AS altas
  FROM truncado
  GROUP BY trimestre
  ORDER BY trimestre DESC
),
axo_ano AS (
  SELECT
    ano,
    SUM(obitos) AS obitos,
    SUM(alta) AS altas
  FROM truncado
  GROUP BY ano
  ORDER BY ano DESC
)

SELECT * FROM axo_trimestre;

```

### 1.0.3 Evolução da ocupação COVID vs ocupação hospitalar

Nesta consulta, calculamos o percentual de leitos ocupados por casos de Covid-19 em relação ao total de leitos ocupados no hospital.

É selecionada a coluna da data de notificação, id do hospital e então o cálculo do percentual, realizando o cálculo “leitos covid” / “leitos ocupados” caso, o valor de “leitos ocupados” seja maior do que 0 e então, é realizada a ordenação por id do hospital e então a data da notificação.

```

-- Evolução da razão entre leitos ocupados por pacientes com COVID e o total de leitos ocupados
CREATE OR MATERIALIZED VIEW evolucao_ocupacao_covid_vs_hospitalar AS
WITH p_ocup AS (
  SELECT
    data_notificacao,
    id_hospital,
    CASE
      WHEN ocupacao_hospitalar > 0 THEN (ocupacao_covid / ocupacao_hospitalar)
      ELSE NULL
    END AS p_ocup_covid

```

```

    FROM registro_ocupacao_final
)
SELECT * FROM p_ocup
ORDER BY id_hospital, data_notificacao;

```

#### 1.0.4 Ocupação máxima de leitos por hospital

Para verificar a não somente a ocupação máxima de leitos atingida, mas também a ocupação máxima por casos de covid e as maiores quantidades de desocupação por obito e por altas, são selecionadas os valores máximos de cada coluna referenciada acima e o id dos hospitais. Para que não tenhamos duas observações com a mesma informação, agrupamos os valores pela id do hospital. Além disto, são selecionadas as observações que não tem os valores de ocupação nulos, ou seja, as ocupações precisam ter valores numéricos para aparecer na análise. Esta visualização oferece uma análise mais objetiva e de fácil entendimento da capacidade de ocupação de cada hospital, e uma possível verificação de sobrecarga do sistema de saúde.

```

-- 7 Máximo de ocupações por hospital
CREATE OR MATERIALIZED VIEW max_ocupacao_hospital AS
WITH ocup_max AS (
    SELECT
        id_hospital,
        MAX(ocupacao_covid) AS ocup_c,
        MAX(ocupacao_hospitalar) AS ocup_h,
        MAX(saida_covid_obitos) AS saida_c_obitos,
        MAX(saida_covid_altas) AS saida_c_altas
    FROM registro_ocupacao_final
    GROUP BY id_hospital
)

SELECT * FROM ocup_max
WHERE ocup_c IS NOT NULL AND ocup_h IS NOT NULL
ORDER BY ocup_h DESC, id_hospital;

```

### 1.1 Observações Finais

Dados são apenas dados até que se tenha um profissional por trás dos bastidores trabalhando:

- Comunicando-se curiosamente com aqueles que mais entendem sobre o negócio ao qual os dados registram os fatos;
- Entendendo o problema com o qual se está lidando, mesmo que nem sempre tudo esteja claro no começo;

- Utilizando tecnicamente ferramentas para tratar, modelar e analisar os dados;
- Interpretando os resultados de forma crítica e contextualizada;
- Compartilhando o conhecimento adquirido de forma clara e objetiva.

Aprender SQL durante o semestre e aplicá-lo praticamente durante este trabalho final foi uma experiência enriquecedora. A linguagem SQL se mostrou uma ferramenta poderosa e completa para trabalhar com dados (ganhando ainda mais o meu apreço), permitindo lidar com grandes volumes de dados. Além disso, este trabalho revelou a importância da organização e estruturação dos dados para facilitar análises futuras.