Universidade Federal do Rio de Janeiro
Instituto de Computação
Professor: Geraldo Xexéo

Relatório - Grupo 5
Ana Clara Monteiro de Oliveira, Andreza Cardoso Santos, Douglas Villalba dos Santos, Jonatas Luis Ramos Simões

Análise de dados sobre a COVID

Questão 1

Para iniciar o trabalho, foi chamada a função "*Unzip files (legacy)*" para baixar o arquivo e descompactar automaticamente salvando em uma pasta local do Knime.
Os dados foram coletados a partir deste link: https://www.saopaulo.sp.gov.br/planosp/simi/dados-abertos/.
Depois foi configurado dentro do nó "*Unzip files (legacy)*" como mostra a figura abaixo:

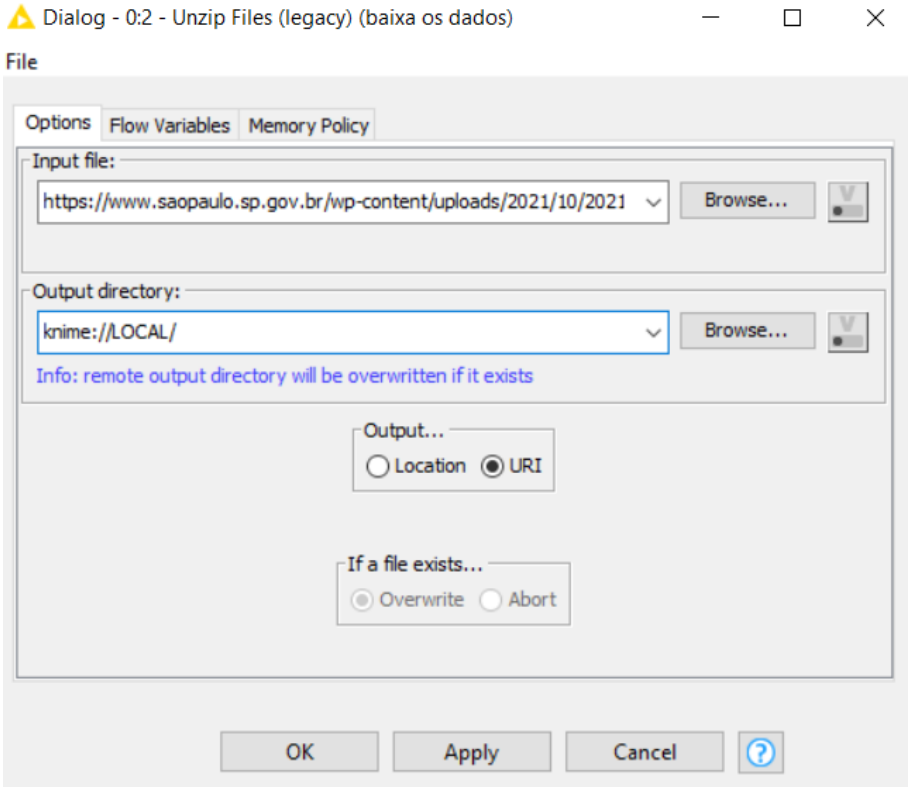

Figura 1 - Configuração do "*Unzip files (legacy)*"

Após esses passos foi utilizado o nó "CSV Reader" para ler o arquivo em CSV descompactado anteriormente, para que assim fosse possível analisar os dados obtidos. Para que o arquivo seja lido sem a necessidade de procurar o arquivo, a configuração do CSV Reader deve estar ajustada para o modo '*Mounpoint*' e '*LOCAL*' onde no campo *File* estará o nome do arquivo. csv.

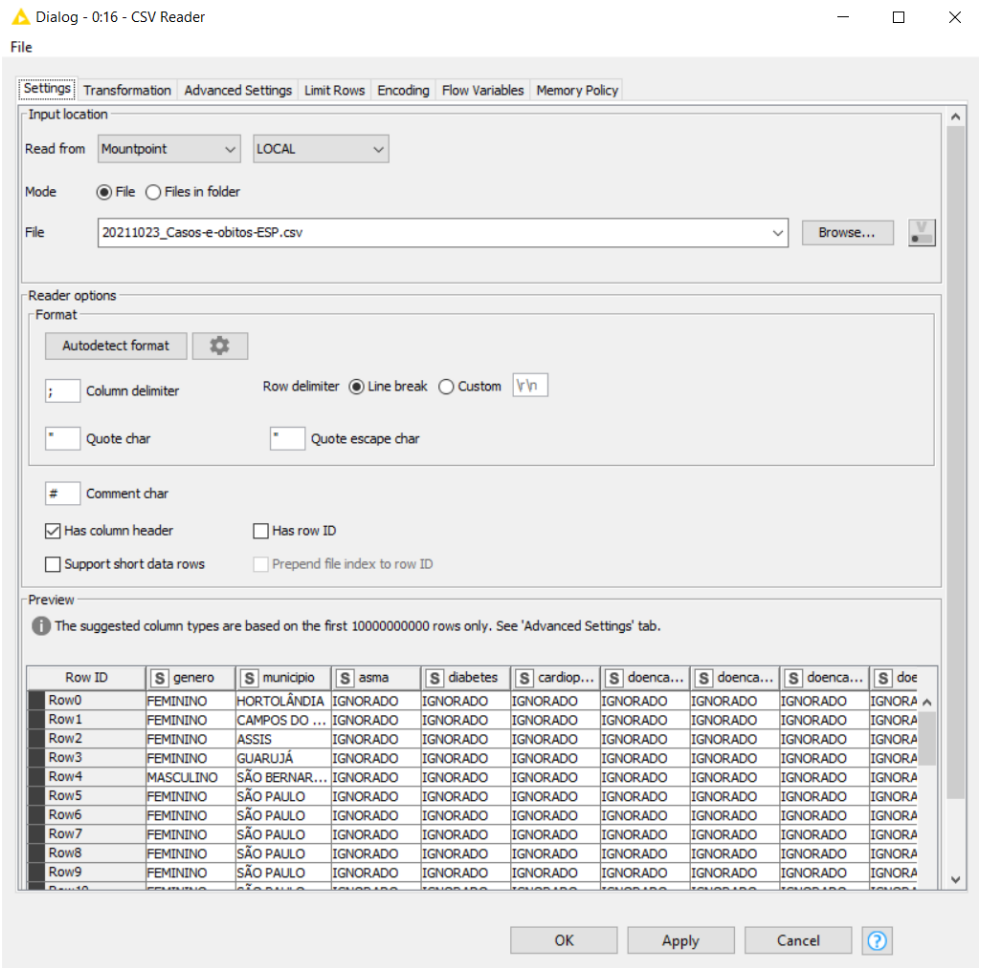

Figura 2 - Configuração do "CSV Reader"

Após as configurações, no campo *'Format'* seleciona-se o '*Autodetect format*' para que ele faça a separação correta do arquivo.

Questão 2

Nesta questão, optou-se por utilizar a ferramenta *Db Diagram* (https://dbdiagram.io/d), para a construção das tabelas do modelo estrela através do código que está na página de anexos. Fizemos o nosso modelo relacional constituído de 3 dimensões e uma tabela fato.

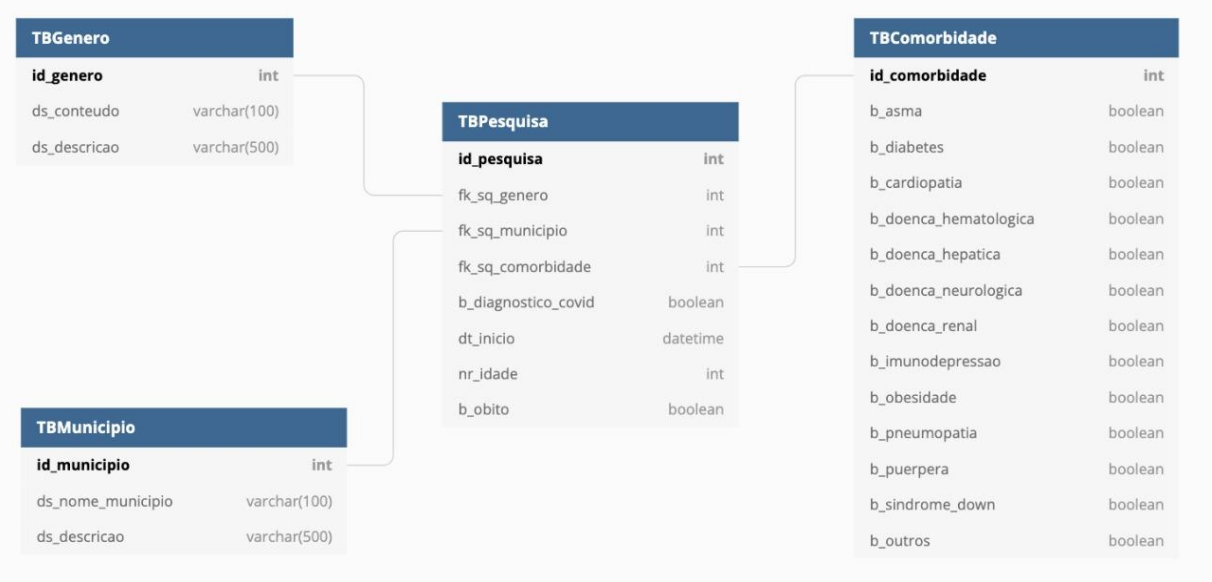

Figura 3 - Modelo dimensional Estrela

Questão 3

Usamos um servidor externo para criar a base de dados no *MySQL* versão 5.7, para que todos os integrantes do grupo pudessem ter acesso a mesma base de dados. Tivemos que usar a versão 5.7, porque um nó usado no *Knime* para carregar os dados não suportava o *MySQL* com versão maior, devido o nó estar obsoleto. Usamos a ferramenta *DataGrip* para criar o banco de dados relacional de acordo com o modelo construído acima (Figura 3). E então geramos o *script* da estrutura do nosso banco que se encontra no anexo [2].

Questão 4

Foi criado um fluxo no *KNIME* para que a carga de dados fosse feita de uma vez só, desde o *download* dos arquivos até o tratamento dos dados.

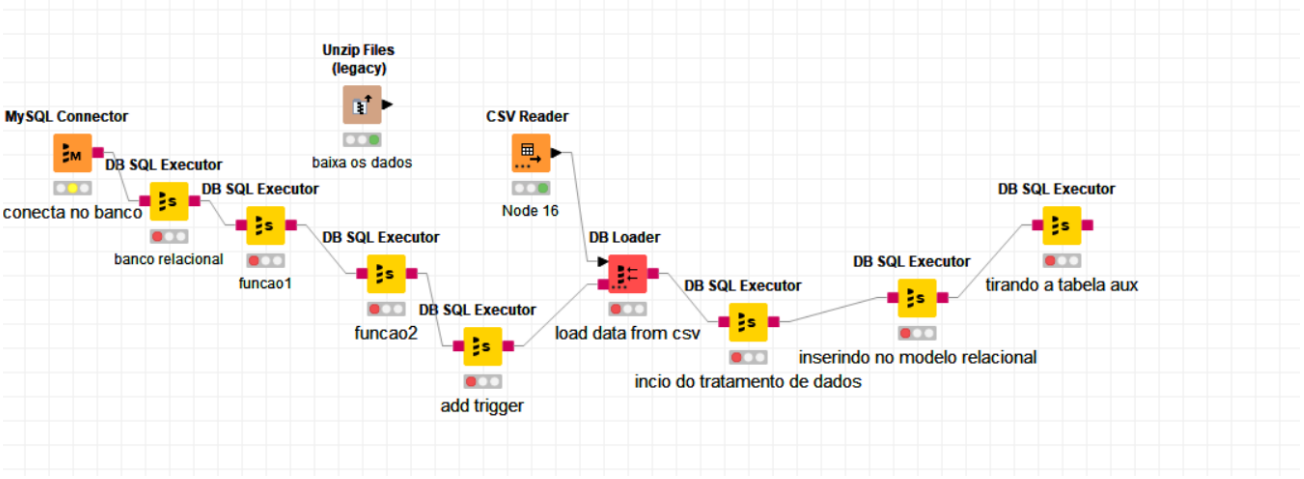

Figura 4 - Fluxo no Knime para a carga dos dados.

Abaixo serão descritos os nós utilizados no fluxo:

- MySQL Connector (Conecta no banco): Primeiro foi adicionado o nó '*MySQL Connector'* onde colocamos as configurações de conexão do banco, sendo elas: IP, porta, usuário e senha:

- DB SQL Executor (banco relacional): Nesse nó foi executado o *script* do anexo 2, onde foi criada o *database*;

- DB SQL Executor (funcao1): Nesse nó foi executado o *script* do anexo 3, função que ajuda no tratamento dos dados;

- DB SQL Executor (funcao2): Nesse nó foi executado o *script* do anexo 4, função que ajuda no tratamento dos dados;

- DB SQL Executor (add trigger): Nesse nó foi executado o *script* do anexo 5, onde a *trigger* é utilizada para auxiliar na inserção de dados;

- DB Loader (load data from csv): Este nó faz a carga dos dados para o MySQL, onde configuramos o banco de dados e a tabela (tabela_aux2). O banco contém duas tabelas auxiliares, para facilitar a inserção de dados no banco. A estrutura da tabela auxiliar 2 está igual ao arquivo CSV, contendo a mesma ordem das colunas e todos os campos são *VARCHAR*, pois assim os dados são melhor otimizados. Já a estrutura da tabela auxiliar 1 também está igual ao arquivo CSV, contendo a mesma ordem das colunas e todos os campos tem o mesmo tipo indicado no dicionário da origem dos dados;

- DB SQL Executor (início do tratamento de dados): Para o início do tratamento, os dados foram separados pelas seguintes tabelas:
    - TBComorbidade: Tabela onde foram inseridos os dados relacionados às comorbidades;
    - TBGenero: Tabela onde foram inseridos os dados relacionados às informações relacionadas ao gênero do paciente;
    - TBMunicipio: Tabela onde foram inseridos os dados relacionados ao município do posto de saúde;
    - TBPesquisa: Tabela onde foram inseridos os dados relacionados às informações relacionadas ao óbito, idade, data de início dos sintomas e confirmação do diagnóstico do COVID-19;

- DB SQL Executor (inserindo no modelo relacional): Será utilizada a *trigger* criada anteriormente para fazer a inclusão dos dados no nosso modelo relacional, de maneira ordenada;

- DB SQL Executor (tirando a tabela aux): Após as inserções e os tratamentos feitos, as tabelas auxiliares (1 e 2) foram deletadas do banco.

É importante frisar que antes da inserção de dados total, a carga de gênero e município foram introduzidas antes dos outros dados, porque essas informações não irão se repetir e sim serem referenciadas. Então

fizemos um *insert* para colocar todos os gêneros, municípios encontrados nos dados inseridos na tabela auxiliar, usando duas *queries* SQL (uma para município e outra para gênero).

Query gênero

```
INSERT INTO
TBGenero(ds_conteudo)
SELECT
   DISTINCT genero
FROM tabela_aux
   WHERE genero is not null;
```

Query município

```
INSERT INTO
TBMunicipio(ds_nome_municipio)
SELECT
   DISTINCT municipio
FROM tabela_aux
   WHERE municipio is not null;
```

Depois fizemos uma *trigger* na tabela de comorbidade para facilitar a inclusão de dados no nosso modelo relacional, então a cada linha da tabela auxiliar que for inserida na tabela de comorbidade, ao mesmo tempo o restante dos dados da tabela auxiliar serão incluídos nas outras dimensões e substituir por foreign key as dimensões já inseridas (gênero e município).

Questão 5

Para essa questão, o grupo optou por utilizar a ferramenta *Tableau* para a realização dos gráficos e tabelas. O banco de dados criado nas questões anteriores foi conectado ao Tableau para que fosse possível utilizar estes dados e assim efetuar as análises.

Foram elaboradas 5 questões para poder analisar os dados do covid-19 dos municípios de São Paulo, que se encontram abaixo.

Pergunta 1: Qual gênero obteve mais óbito?

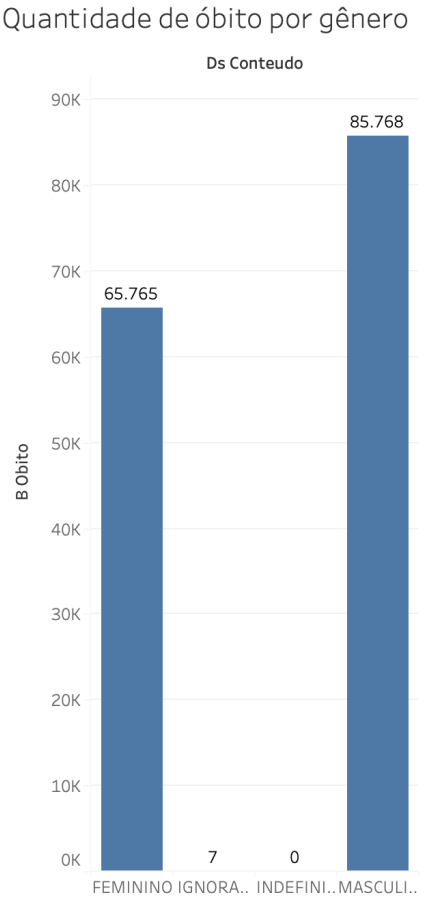

Figura 5 - Gráfico de óbito por gênero

Pode-se observar que o gênero masculino foi mais afetado pelo covid-19 obtendo um total de 85.768 óbitos durante os anos de 2020 e 2021.

Pergunta 2: Quais os 10 municípios mais atingidos?

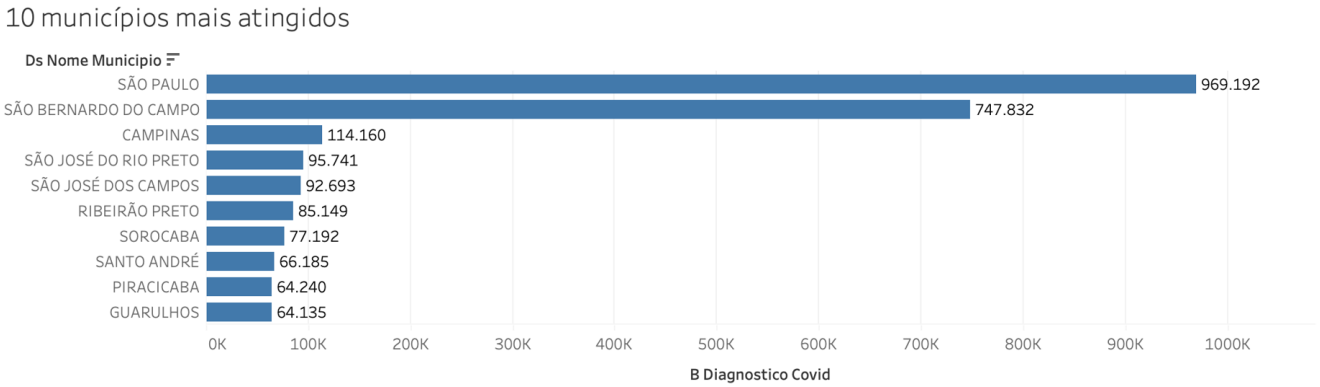

Figura 6 - Gráfico dos 10 municípios mais atingidos

Percebe-se por este gráfico que dentre os 10 municípios mais atingidos, São Paulo e São Bernardo do Campo lideram com os maiores índices de diagnósticos positivos para Covid-19.

Pergunta 3: Qual a idade obteve mais óbito?

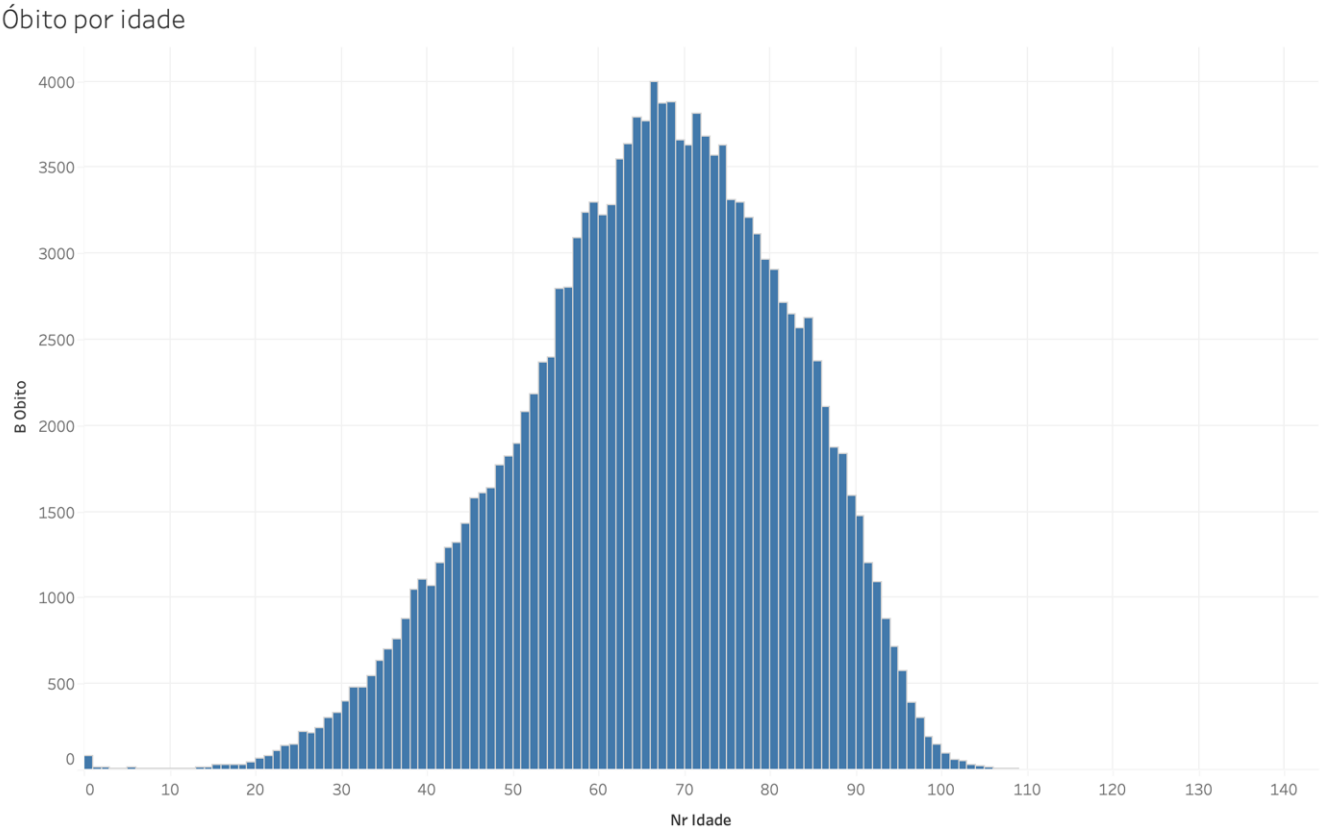

Figura 7 - Gráfico da relação de idade por óbito

Analisando o gráfico acima, observa-se a concentração de óbitos na faixa dos 60-70 anos, idade que é considerada com grupo de risco.

Pergunta 4: Qual comorbidade obteve mais óbitos?

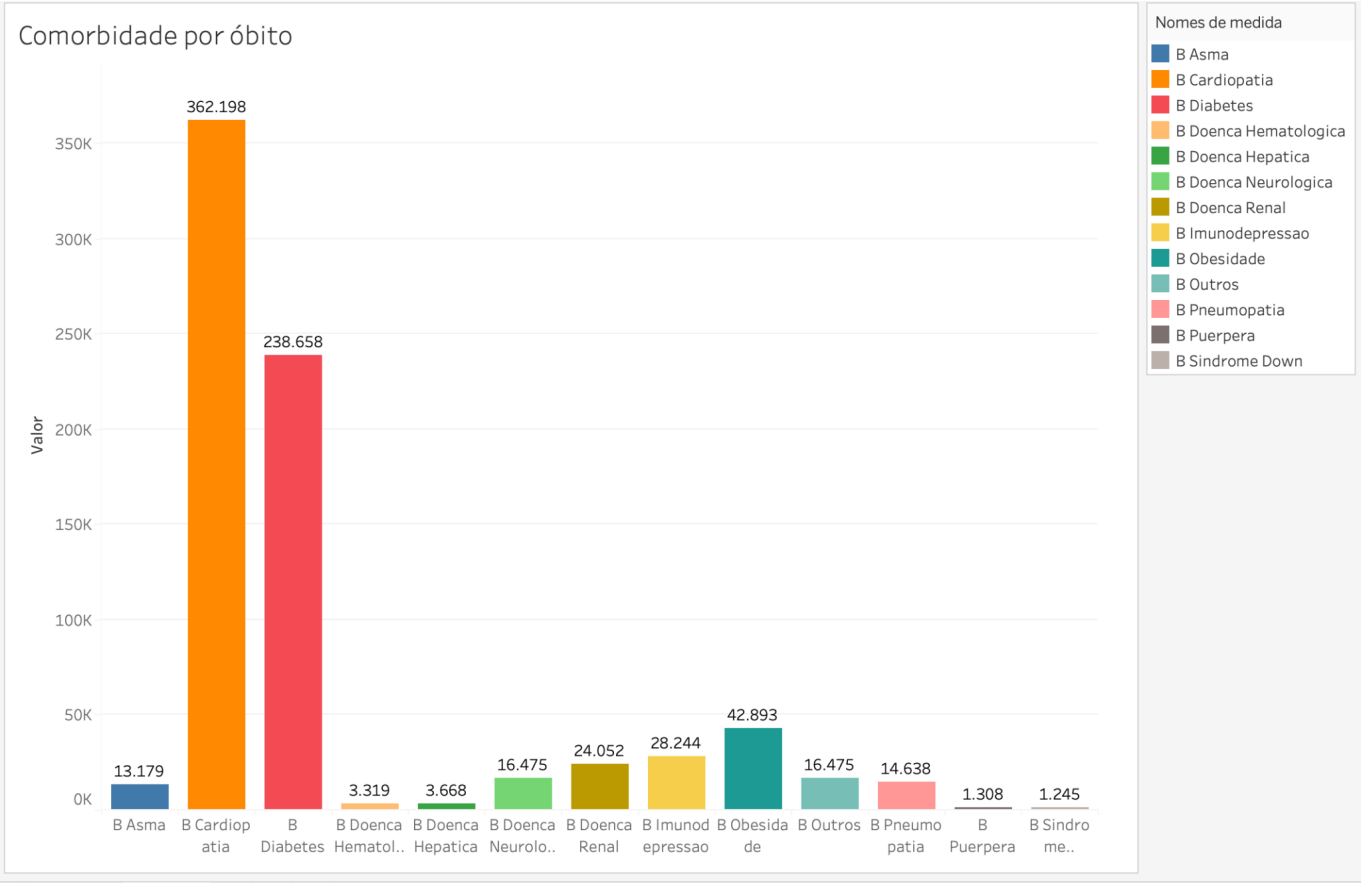

Figura 8 - Gráfico do tipo de comorbidade relacionado ao óbito.

Pelo gráfico acima, nota-se que pessoas portadoras de Cardiopatia obteve um alto índice de óbito, seguida pelas que têm diabetes, como comorbidade.

Pergunta 5: Qual idade obteve mais diagnósticos de confirmados ?

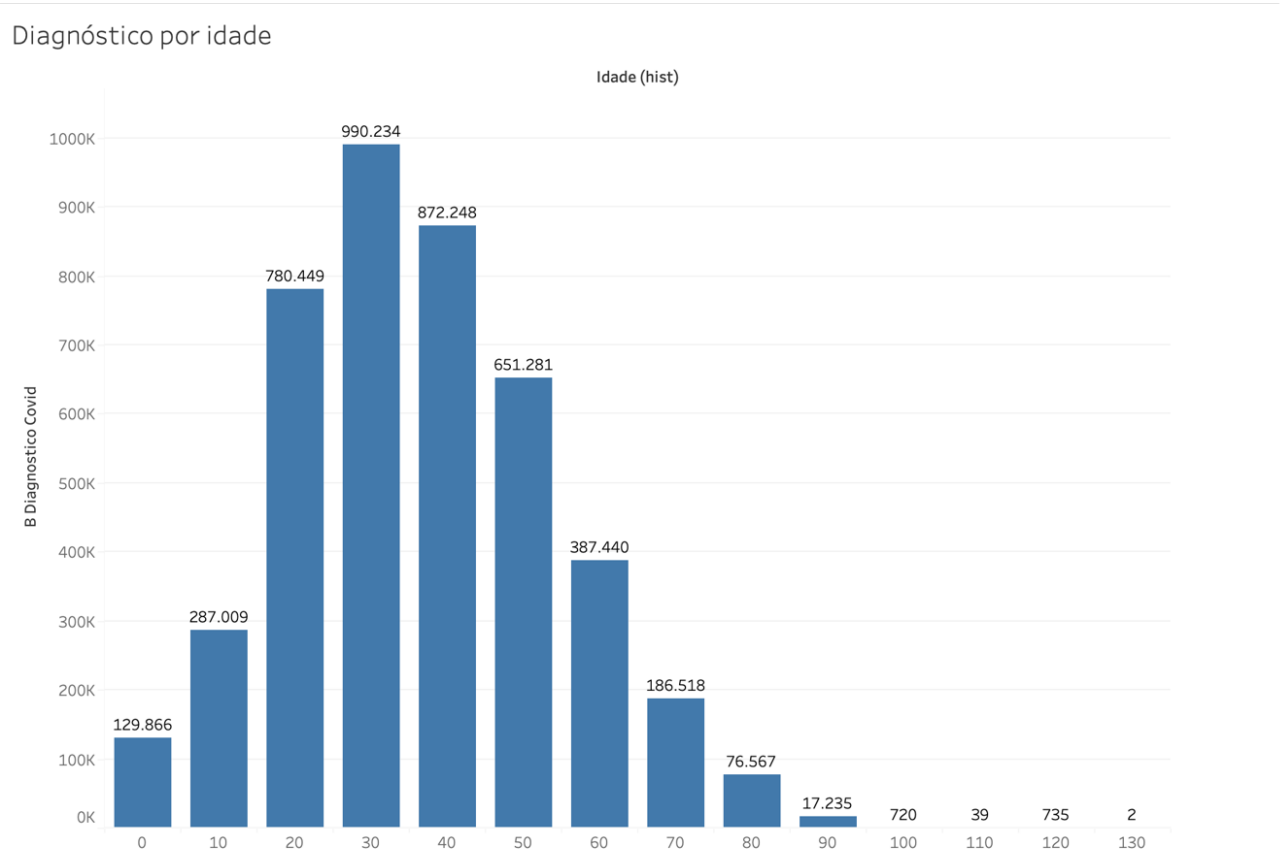

Figura 9 - Gráfico da relação de idade por diagnósticos de confirmados.

Observando o gráfico, pode-se analisar que as pessoas adultas, a partir dos 30 anos, foram as mais atingidas pelo Covid-19, sendo nesse setor dos 30-40 anos, o maior índice de casos ocorridos.

Questão 6

Para o aprendizado de máquina, decidimos colocar como opção a quantidade de óbitos por comorbidade, no caso foi escolhido a comorbidade ‘asma’. Primeiro foi conectado o banco de dados, depois foi feita uma consulta em sql (Figura 10**)** com as variáveis utilizadas, e então implementar os nós para o aprendizado de máquina. Para a consulta botamos um limite de apenas 20 mil linhas, pois o grupo não possui máquinas com o processamento adequado para gerar os resultados com os dados completos no *KNIME*, que são cerca de 4 milhões de linhas.

SQL Statement

```sql
select
        TC.b_asma
        ,TP.dt_inicio
        ,TP.b_diagnostico_covid
        ,TP.b_obito
FROM TBComorbidade TC
    INNER JOIN TBPesquisa TP
        on TC.id_comorbidade = TP.fk_sq_comorbidade
where 1=1
  and TP.b_diagnostico_covid = 1
  and TP.b_obito = 1
  and TC.b_asma is not null
order by TP.dt_inicio DESC
limit 20000;
```

Figura 10 - Consulta em SQL

O nó *'X-partitioner'* foi utilizado para definir quantas iterações seriam feitas e optou-se por colocar duas iterações, mínimo exigido na questão. Em seguida, este nó foi conectado em mais dois, *'SVM Learner'* que serve para treinar uma máquina de vetores, e *'SVM Predictor'* que serve para prever a saída dos valores. As configurações dos respectivos nós são as seguintes:

- ***SVM Learner*:**

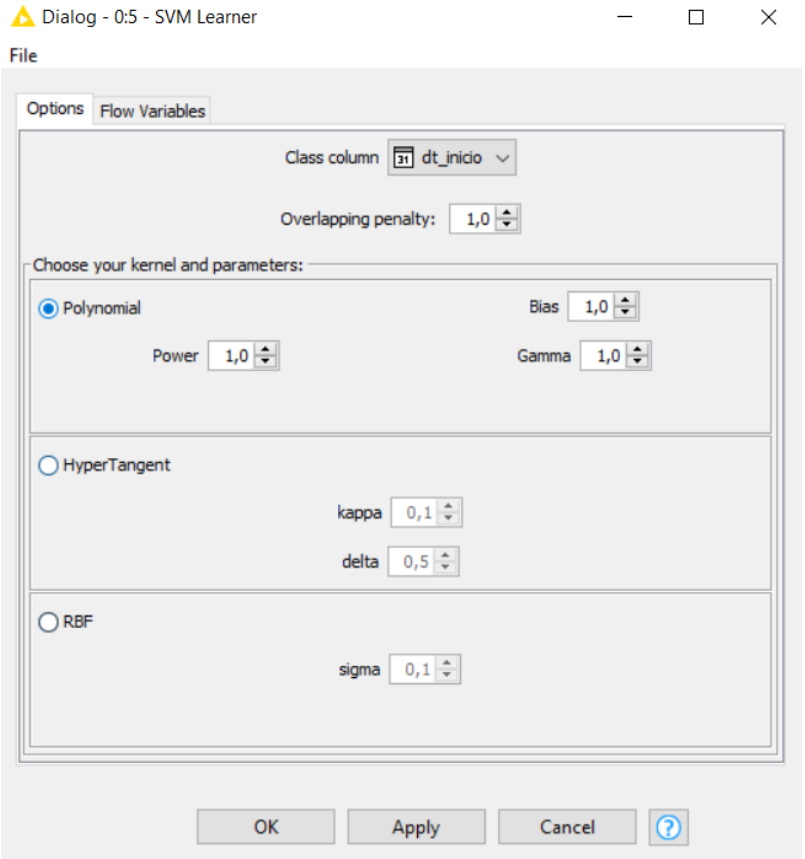

Figura 11- Configuração do *SVM Learner*

- ***SVM Predictor*:** Não houve necessidade de configurá-lo.

.

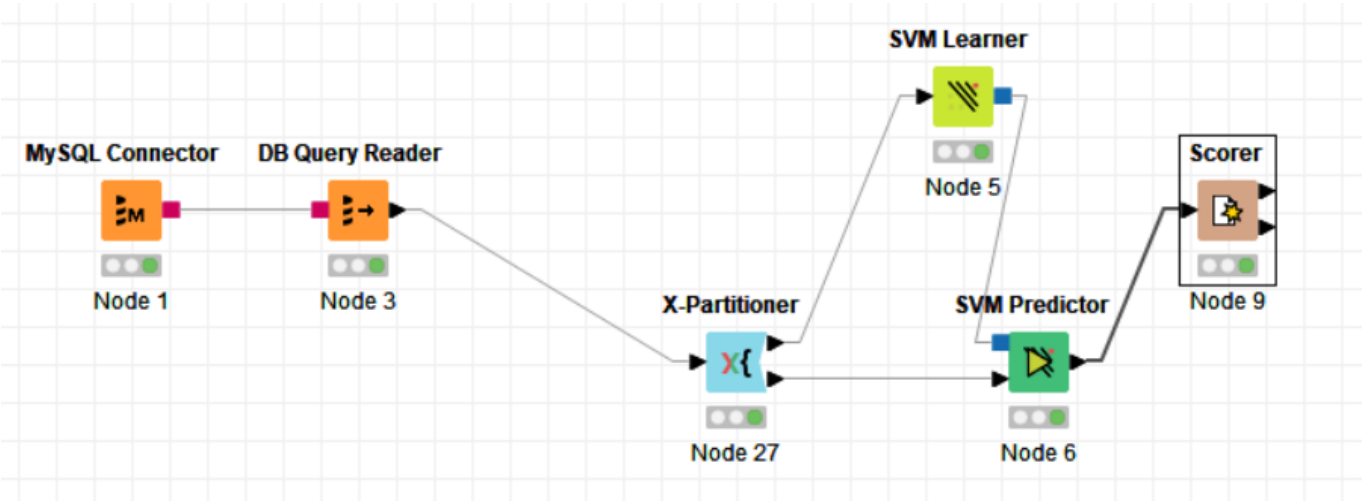

Figura 12 - Fluxo do Knime para o aprendizado

Após aplicar os nós, foram gerados os resultados abaixo (figura 13). Mesmo com duas tentativas de aprendizado a precisão e acurácia tiveram um valor de 5,1%.
Inicialmente o grupo rodou os nós com 1000 linhas, porém os valores da precisão e acurácia foram muito inferiores. Depois de algumas tentativas, aumentando a quantidade de linhas até o limite suportado pela máquina, chegamos a conclusão que caso fosse feita a consulta com o número total de linhas, 4 milhões, a tendência é que o valor da precisão e acurácia chegasse no valor esperado.

Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	Recall	Precision	Sensitivity	Specificity	F-meas...	Accuracy	Cohen'...
true	515	9485	0	0	1	0.051	1	0	0.098	?	?
false	0	0	515	9485	0	?	0	1	?	?	?
Overall	?	?	?	?	?	?	?	?	?	0.051	0

Figura 13 - Resultado da acurácia e precisão.

Questão 7

- Db Diagram: https://dbdiagram.io/d - Essa ferramenta foi escolhida por conta da sua facilidade em montar o modelo dimensional.
- DataGrip: https://www.jetbrains.com/datagrip/ - Essa ferramenta foi escolhida por ser mais intuitiva que o MySQL workbench.
- *Knime*: https://www.knime.com/ - Essa ferramenta foi utilizada para fazer o download automático dos dados e para o aprendizado de máquina.
- Tableau: https://www.tableau.com/pt-br - Essa ferramenta é muito utilizada para a geração de gráficos e tabelas.
- GitHub: https://github.com/simoesjonatas/DataWarehouse_grupo5 - O GitHub foi utilizado para colocar o repositório dos dados analisados.
- Google Cloud: https://cloud.google.com - Foi utilizado para alocar o banco de dados MySQL 5.7.

ANEXO

1. Script para o modelo dimensional

```
//// -- Tables and References

// Creating tables
Table TBGenero as U {
  id_genero int [pk, increment, unique] // auto-increment
  ds_conteudo varchar(100)
  ds_descricao varchar(500)
}

Table TBMunicipio {
  id_municipio int [pk, increment, unique]
  ds_nome_municipio varchar(100) [unique]
  ds_descricao varchar(500)
 }

Table TBComorbidade {
  id_comorbidade int [pk, increment, unique]
  b_asma boolean [null]
  b_diabetes boolean [null]
  b_cardiopatia boolean [null]
  b_doenca_hematologica boolean [null]
  b_doenca_hepatica boolean [null]
  b_doenca_neurologica boolean [null]
  b_doenca_renal boolean [null]
  b_imunodepressao boolean [null]
  b_obesidade boolean [null]
  b_pneumopatia boolean [null]
  b_puerpera boolean [null]
  b_sindrome_down boolean [null]
  b_outros boolean [null]
}

Table TBPesquisa {
  id_pesquisa int [pk, increment, unique] // primary key
  fk_sq_genero int [null]
  fk_sq_municipio int [null]
  fk_sq_comorbidade int [null]
  b_diagnostico_covid boolean
  dt_inicio datetime [null]
```

```
  nr_idade int [not null]
  b_obito boolean [not null]
}
//composite foreign key
Ref: TBPesquisa.(fk_sq_genero) > TBGenero.(id_genero)
Ref: TBPesquisa.(fk_sq_municipio) > TBMunicipio.(id_municipio)
Ref: TBPesquisa.(fk_sq_comorbidade) > TBComorbidade.(id_comorbidade)
```

Obtendo-se o seguinte Modelo Estrela, como visto na imagem abaixo, com uma tabela fato (TBPesquisa) e três dimensões (TBGenero, TBMunicipio e TBComorbidade).

2. Estrutura do banco

```
-- criando database
CREATE DATABASE IF NOT EXISTS dw_pesquisa;
USE dw_pesquisa;

-- criando as tabelas do nosso modelo relacional
create table if not exists TBComorbidade
(
	b_outros tinyint(1) null,
	b_sindrome_down tinyint(1) null,
	b_puerpera tinyint(1) null,
	b_pneumopatia tinyint(1) null,
	b_obesidade tinyint(1) null,
	b_imunodepressao tinyint(1) null,
	b_doenca_renal tinyint(1) null,
	b_doenca_neurologica tinyint(1) null,
	b_doenca_hepatica tinyint(1) null,
	b_doenca_hematologica tinyint(1) null,
	b_cardiopatia tinyint(1) null,
	b_diabetes tinyint(1) null,
	b_asma tinyint(1) null,
	id_comorbidade int auto_increment,
	constraint TBComorbidade_id_comorbidade_uindex
		unique (id_comorbidade)
);

alter table TBComorbidade
	add primary key (id_comorbidade);

create table if not exists TBGenero
(
	ds_descricao varchar(500) null,
```

```
	ds_conteudo varchar(100) not null,
	id_genero int auto_increment,
	constraint TBGenero_ds_conteudo_uindex
		unique (ds_conteudo),
	constraint TBGenero_id_genero_uindex
		unique (id_genero)
);

alter table TBGenero
	add primary key (id_genero);

create table if not exists TBMunicipio
(
	ds_descricao varchar(500) null,
	ds_nome_municipio varchar(100) not null,
	id_municipio int auto_increment,
	constraint TBMunicipio_id_municipio_uindex
		unique (id_municipio)
);

alter table TBMunicipio
	add primary key (id_municipio);

create table if not exists TBPesquisa
(
	b_obito tinyint(1) null,
	nr_idade int null,
	dt_inicio date null,
	b_diagnostico_covid tinyint(1) null,
	fk_sq_comorbidade int null,
	fk_sq_municipio int null,
	fk_sq_genero int null,
	id_pesquisa int auto_increment,
	constraint TBPesquisa_id_pesquisa_uindex
		unique (id_pesquisa),
	constraint TBPesquisa_TBComorbidade_id_comorbidade_fk
		foreign key (fk_sq_comorbidade) references TBComorbidade (id_comorbidade),
	constraint TBPesquisa_TBGenero_id_genero_fk
		foreign key (fk_sq_genero) references TBGenero (id_genero),
	constraint TBPesquisa_TBMunicipio_id_municipio_fk
		foreign key (fk_sq_municipio) references TBMunicipio (id_municipio)
);

alter table TBPesquisa
```

```
        add primary key (id_pesquisa);

create table if not exists tabela_aux
(
        genero varchar(100) null,
        municipio varchar(100) null,
        asma varchar(100) null,
        diabetes varchar(100) null,
        cardiopatia varchar(100) null,
        doenca_hematologica varchar(100) null,
        doenca_hepatica varchar(100) null,
        doenca_neurologica varchar(100) null,
        doenca_renal varchar(100) null,
        imunodepressao varchar(100) null,
        obesidade varchar(100) null,
        pneumopatia varchar(100) null,
        puerpera varchar(100) null,
        sindrome_down varchar(100) null,
        outros varchar(100) null,
        dig_covid varchar(100) null,
        data_inicio varchar(100) null,
        idade int null,
        obito tinyint(1) null,
        id_aux int auto_increment
                primary key
);

create table if not exists tabela_aux2
(
    genero varchar(100) null,
        municipio varchar(100) null,
        asma varchar(100) null,
        diabetes varchar(100) null,
        cardiopatia varchar(100) null,
        doenca_hematologica varchar(100) null,
        doenca_hepatica varchar(100) null,
        doenca_neurologica varchar(100) null,
        doenca_renal varchar(100) null,
        imunodepressao varchar(100) null,
        obesidade varchar(100) null,
        pneumopatia varchar(100) null,
        puerpera varchar(100) null,
        sindrome_down varchar(100) null,
        outros varchar(100) null,
```

```
        dig_covid varchar(100) null,
        data_inicio varchar(100) null,
        idade varchar(100) null,
        obito varchar(100) null
);
```

3. Criando uma função para ajudar no tratamento dos dados

```
create  function DW_EXTRACT_CONFIRMADO(objeto varchar(100)) returns text
BEGIN
      IF objeto in ('CONFIRMADO') THEN
         RETURN 1;
      ELSE
         RETURN 0;
      END IF;
END;
```

4. Criando uma função para ajudar no tratamento dos dados

```
create  function DW_EXTRACT_TXT(objeto varchar(100)) returns text
BEGIN
      #DECLARE info varchar(1000);
      IF objeto IN (NULL,'NULL','null','IGNORADO') THEN
            RETURN NULL;
      ELSEIF objeto in ('SIM') THEN
         RETURN 1;
      ELSEIF objeto in ('NÃO','nao','Não','Nao') THEN
         RETURN 0;
      ELSE
            RETURN NULL;
      END IF;
END;
```

5. Criando trigger para fazermos a inserção de dados no nosso modelo relacional

```
create  trigger TBComorbidade_TR_ON_INSERT
        after insert
        on TBComorbidade
        for each row
        BEGIN

     INSERT  INTO  TBPesquisa(FK_SQ_GENERO,  FK_SQ_MUNICIPIO,  FK_SQ_COMORBIDADE,
B_DIAGNOSTICO_COVID, DT_INICIO, NR_IDADE, B_OBITO)
   SELECT (SELECT id_genero FROM TBGenero WHERE ds_conteudo = genero),
        (SELECT id_municipio FROM TBMunicipio WHERE ds_nome_municipio = municipio ),
```

```
        NEW.id_comorbidade,
        DW_EXTRACT_CONFIRMADO(dig_covid),
        str_to_date(data_inicio, "%d/%m/%Y"),
        idade,
        obito
    FROM tabela_aux WHERE id_aux = NEW.id_comorbidade;

END;
```