Teste técnico PAGOZ

Lucas Cardoso de Menezes

12/05/2024

Contexto:

O time de vendas de uma empresa do setor de meios de pagamentos deseja levantar algumas informações das transações realizadas. Várias tabelas em formatos diferentes foram enviadas, será necessário que o time de dados adeque as informações e agregue os dados a tabela principal “transacoes.csv” e realize algumas análises para o time de vendas.

Proposta:

Através dos dados diponibilizados, realizar as análises utilizando script R, explorando as informações e demonstrando a análise para o usuário.

Análises obrigatórias solicitadas:

1.1. Qual estabelecimento movimentou o cógido api CAOD9ND23H4B7ZCMGWXAN47Z8KEWAF5W?

1.2. Quantas vendas em cartão de crédito foram realizadas entre a data 2022-01-18 e 2023-02-01?

2. Quais são os clientes com maior valor de faturamento? (Identificados na coluna “estabelecimento”)
3. Quantas máquinas cada cliente possui?
4. Qual o valor total em “eventos” negativos?
5. Qual a porcentagem de vendas com “Leitor de chip e senha” em relação a “Venda pela Maquininha”? Em número de transações e valor de faturamento total.
6. Precisamos de um gráfico que demonstre o faturamento total por parcelas.
7. Gráfico com a razão de transações débito, crédito e parcelado.

Preparando o ambiente para à análise:

Para realizar essa análise contaremos com o uso de algumas bibliotecas do R, os principais serão o *dplyr* e o *RSQLite*. Um para auxiliar na manipulação das tabelas e o outro será utilizado para a montagem dos bancos de dados.

Os pacotes estão disponíveis em:

- dplyr: https://cran.r-project.org/web/packages/dplyr/index.html (https://cran.r-project.org/web/packages/dplyr/index.html)
- RSQLite: https://cran.r-project.org/web/packages/RSQLite/index.html (https://cran.r-project.org/web/packages/RSQLite/index.html)
- data.table: https://cran.r-project.org/web/packages/data.table/index.html (https://cran.r-project.org/web/packages/data.table/index.html)
- readxl: https://cran.r-project.org/web/packages/readxl/index.html (https://cran.r-project.org/web/packages/readxl/index.html)

- plotly: https://cran.r-project.org/web/packages/plotly/index.html (https://cran.r-project.org/web/packages/plotly/index.html)
- stringr: https://cran.r-project.org/web/packages/stringr/index.html (https://cran.r-project.org/web/packages/stringr/index.html)

Ou utilizando os comandos:

```
# install.packages("dplyr")
# install.packages("RSQLite")
# install.packages("data.table")
# install.packages("readxl")
# install.packages("stringr")
# install.packages("plotly")
```

Vamos utilizar o *libary* para chamar os pacotes após a instalação:

```
library(data.table)
library(readxl)
library(dplyr)
library(stringr)
library(RSQLite)
library(plotly)
```

Para finalizarmos a preparação do ambiente devemos informar o local onde se encontra os dados de entrada:

```
#setwd(dir = "...Tabelas/")
```

#Programação em R

1. Leitura e montagem dos dados

Para leitura do dados utilizaremos o pacote data.table, que possuí a função *fread()*, uma das funções mais perfomáticas para leitura de arquivos csv/txt, utilizaremos também o pacote reaxl para ler os arquivos de extensão xls/xlsx.

```
#Dados auxilaires ----

leitor <- data.table::fread(
  file = paste0(getwd(), "/Tabelas/leitor.csv"), sep = ",", encoding = "UTF-8"
) %>%
  dplyr::rename("descricao_leitor" = "descricao")

head(leitor)
```

```
##    # leitor  descricao_leitor
##       <int>            <char>
## 1:        0    Operação conta
## 2:        1 Leitor de crédito
## 3:       10               BTW
## 4:       11          BTW E275
## 5:       12    Maquininha PRO
## 6:       14              D150
```

```
captura <- data.table::fread(
  file = paste0(getwd(), "/Tabelas/captura.csv"), sep = ",", encoding = "UTF-8"
) %>%
  dplyr::rename("descricao_captura" = "descricao")

head(captura)
```

```
##    # meio_captura descricao_captura
##             <int>            <char>
## 1:              1              Chip
## 2:              2             Tarja
## 3:              3    Não presencial
## 4:              4               TEF
## 5:              5           QR Code
```

```
pagamento <- data.table::fread(
  file = paste0(getwd(), "/Tabelas/pagamento.csv"), encoding = "Latin-1"
) %>%
  dplyr::rename("descricao_pagamento" = "descricao")

head(pagamento)
```

```
##    # meio_pagamento        descricao_pagamento
##               <int>                     <char>
## 1:                0            Operação Conta
## 2:                1             Débito Online
## 3:               11                       Pix
## 4:               12             Carne Credito
## 5:               13              Carne Debito
## 6:               14 Cartao de Credito Pre-Pago
```

```
status_pagamento <- data.table::fread(
  file = paste0(getwd(), "/Tabelas/status_pagamento.csv"), encoding = "Latin-1"
) %>%
  dplyr::rename("descricao_status_pagamento" = "descricao")

head(status_pagamento)
```

```
##    # status_pagamento descricao_status_pagamento
##                 <int>                      <char>
## 1:                 1                   Agendado
## 2:                 2                       Pago
## 3:                 3                  Disponível
## 4:                 4 Disponível por Antecipação
## 5:                 5                  Solicitado
```

```
canal_entrada <- readxl::read_xls(
  path = paste0(getwd(), "/Tabelas/canal_entrada.xls")
) %>%
  dplyr::rename("descricao_canal_entrada" = "descricao")

head(canal_entrada)
```

```
## # A tibble: 6 × 2
##   `# canal_entrada` descricao_canal_entrada
##   <chr>             <chr>
## 1 <NA>              Operação Conta
## 2 AP                Aplicativo
## 3 LK                Link
## 4 MD                Venda digitada
## 5 ME                Venda pela Maquininha
## 6 MP                Venda com leitor de chip e senha
```

```
evento <- readxl::read_xlsx(
  path = paste0(getwd(), "/Tabelas/evento.xlsx")
) %>%
  dplyr::rename("descricao_evento" = "descricao")

head(evento)
```

```
## # A tibble: 6 × 3
##   `# tipo_evento` descricao_evento            sinal
##             <dbl> <chr>                       <chr>
## 1               1 Venda ou Pagamento          +
## 2              10 Encerramento Disputa        +
## 3              11 Abertura Pré-Chargeback     -
## 4              12 Encerramento Pré-Chargeback +
## 5              14 Agendamento Sque CIP        -
## 6              15 Saque Aprovado              <NA>
```

```
#Dado principal ----

transacoes <- data.table::fread(
  file = paste0(getwd(), "/Tabelas/transacoes.csv"), encoding = "Latin-1"
)

colnames(transacoes)
```

```
##  [1] "# movimento_api_codigo"   "tipo_registro"
##  [3] "estabelecimento"          "data_inicial_transacao"
##  [5] "data_venda_ajuste"        "hora_venda_ajuste"
##  [7] "tipo_evento"              "tipo_transacao"
##  [9] "numero_serie_leitor"      "codigo_transacao"
## [11] "codigo_venda"             "valor_total_transacao"
## [13] "valor_parcela"            "pagamento_prazo"
## [15] "plano"                    "parcela"
## [17] "quantidade_parcela"       "data_prevista_pagamento"
## [19] "taxa_parcela_comprador"   "tarifa_boleto_compra"
## [21] "valor_original_transacao" "taxa_parcela_vendedor"
## [23] "taxa_intermediacao"       "tarifa_intermediacao"
## [25] "tarifa_boleto_vendedor"   "taxa_rep_aplicacao"
## [27] "valor_liquido_transacao"  "status_pagamento"
## [29] "meio_pagamento"           "instituicao_financeira"
## [31] "canal_entrada"            "leitor"
## [33] "meio_captura"             "num_logico"
## [35] "nsu"                      "cartao_bin"
## [37] "cartao_holder"            "codigo_autorizacao"
## [39] "codigo_cv"                "date_time"
```

Após ler cada tabela auxiliar, também utilizamos a função *rename()* para renomear as coluna descrição de cada uma delas para um nome mais especifico, com o intuito de futuramente ao realizar a junção das tabelas auxilaires com a tabela princiapal não haver confusão com as nomenclaturas dos dados.

Traçando um paralelo com o processo de leitura dos arquivos, via função de exportação do Excel, observa-se que, em comparação com as leituras realizadas via as funções supracitadas, não tivemos que nos preocupar com ajustes do separadores e delimitadores que seriam necessárias para que o dado estivese disponível para uso sem erro nas informações da coluna. Já que as próprias funções realizam esses ajustes de forma nativa.

Após a leitura de todas as tabelas, vamos realizar a junção dos dados pelas colunas de chave primária de cada tabela:

```
#Join entre as tabelas ----

transacoes <- transacoes %>%
  dplyr::left_join(leitor, by = c("leitor" = "# leitor")) %>%
  dplyr::left_join(captura, by = c("meio_captura" = "# meio_captura")) %>%
  dplyr::left_join(pagamento, by = c("meio_pagamento" = "# meio_pagamento")) %>%
  dplyr::left_join(status_pagamento, by = c("status_pagamento" = "# status_pagamento"))
%>%
  dplyr::left_join(canal_entrada, by = c("canal_entrada" = "# canal_entrada")) %>%
  dplyr::left_join(evento, by = c("tipo_evento" = "# tipo_evento"))
```

Utilizando a função *left_join()*, é possível juntar as tabelas auxiliares com a tabela princiapl através das colunas chave. Se observarmos agora, a tabela transações já contém os dados agregados das tabelas auxiliares:

```
colnames(transacoes)
```

```
##  [1] "# movimento_api_codigo"     "tipo_registro"
##  [3] "estabelecimento"            "data_inicial_transacao"
##  [5] "data_venda_ajuste"          "hora_venda_ajuste"
##  [7] "tipo_evento"                "tipo_transacao"
##  [9] "numero_serie_leitor"        "codigo_transacao"
## [11] "codigo_venda"               "valor_total_transacao"
## [13] "valor_parcela"              "pagamento_prazo"
## [15] "plano"                      "parcela"
## [17] "quantidade_parcela"         "data_prevista_pagamento"
## [19] "taxa_parcela_comprador"     "tarifa_boleto_compra"
## [21] "valor_original_transacao"   "taxa_parcela_vendedor"
## [23] "taxa_intermediacao"         "tarifa_intermediacao"
## [25] "tarifa_boleto_vendedor"     "taxa_rep_aplicacao"
## [27] "valor_liquido_transacao"    "status_pagamento"
## [29] "meio_pagamento"             "instituicao_financeira"
## [31] "canal_entrada"              "leitor"
## [33] "meio_captura"               "num_logico"
## [35] "nsu"                        "cartao_bin"
## [37] "cartao_holder"              "codigo_autorizacao"
## [39] "codigo_cv"                  "date_time"
## [41] "descricao_leitor"           "descricao_captura"
## [43] "descricao_pagamento"        "descricao_status_pagamento"
## [45] "descricao_canal_entrada"    "descricao_evento"
## [47] "sinal"
```

2. Análise e resoluções das questões

Qual estabelecimento movimentou o cógido api CAOD9ND23H4B7ZCMGWXAN47Z8KEWAF5W?

Para resolução dessa questão, basta filtrarmos a coluna “# movimento_api_codigo” utilizando a função *subset*, buscando pelo código passado na questão, selecionar as colunas “# movimento_api_codigo” e “estabelecimento” e por fim imprimir o resultado do filtro.

```
questao_um <- transacoes %>%
  subset(`# movimento_api_codigo` == "CAOD9ND23H4B7ZCMGWXAN47Z8KEWAF5W") %>%
  dplyr::select(`# movimento_api_codigo`, estabelecimento) %>%
  print()
```

```
##              # movimento_api_codigo estabelecimento
##                              <char>           <int>
## 1: CAOD9ND23H4B7ZCMGWXAN47Z8KEWAF5W       296956278
```

Quantas vendas em cartão de crédito foram realizadas entre a data 2022-01-18 e 2023-02-01?

Para chegar nesse quantitativo, primeiro iremos criar uma nova coluna na tabela, denominada “data_venda_ajuste_num”, que irá receber os dados da coluna “data_venda_ajuste”, porém retirando o caractere traço (-). Para isso, utilizamos a função *str_remove_all* que remove strings de um determinando campo. Em seguida, bastou filtrar as vendas que ocoreram entre os périodos solicitados e que eram do tipo

“Cartão de Crédito”, por fim realizar a contagem de linhas retornadas.

```
questao_um_ <- transacoes %>%
  dplyr::mutate(
    data_venda_ajuste_num = as.numeric( str_remove_all( data_venda_ajuste, "-" ) )
    ) %>%
  subset(
    data_venda_ajuste_num > 20220117 & data_venda_ajuste_num < 20230131 & descricao_pagame
nto == "Cartão de Crédito"
  ) %>%
  dplyr::count() %>% print()
```

```
##        n
##    <int>
## 1:  1356
```

Quais são os clientes com maior valor de faturamento? (Identificados na coluna “estabelecimento”)

Para isso, agrupamos os dados pela coluna “estabelecimento” utilizando a função *group_by*, criamos uma nova coluna que seria a soma da coluna “valor_total_transacao”, como os dados estão agrupados será realizada a soma por cada estabelecimento, depois selecionamos apenas as colunas pertinentes para o resultado final e ordenamos a base de forma decrescente utilizando a função *arrange*, com o parâmetro “desc()”, pegamos somente os primeiros dados (os maiores faturamentos) e imprimimos na tela o resultado.

```
questao_dois <- transacoes %>%
  dplyr::group_by(estabelecimento) %>%
  dplyr::mutate(faturamento_por_estabelecimento = sum(valor_total_transacao)) %>%
  dplyr::select(estabelecimento, faturamento_por_estabelecimento) %>%
  dplyr::arrange(desc(faturamento_por_estabelecimento)) %>%
  head() %>% print()
```

```
## # A tibble: 6 × 2
## # Groups:   estabelecimento [5]
##   estabelecimento faturamento_por_estabelecimento
##             <int>                           <dbl>
## 1       320906686                          11156.
## 2       313696580                           6013.
## 3       122268884                           5900
## 4       333227944                           5830
## 5       335202294                           5230
## 6       335202294                           5230
```

Quantas máquinas cada cliente possui?

Está questão que antes era um desafio operacional que gastava algum tempo para ser realizada, pode ser respondida de forma simples utilizando o código a seguir.

Primeiro, selecionamos apenas as colunas: “estabelecimento” e “descricao_leitor”, após isso, utilizando a função *unique()* removemos as repetições na base, assim teremos os estabelecimentos com uma unica

ocorrência por tipo de maquininha. Agrupamos por estabelecimentos e criamos uma nova coluna que terá o número de maquininhas que cada cliente possuí ou o *length()* do grupo. E imprimos na tela o resultado.

Para ter algo mais enxuto, seleciono apenas as colunas: “estabelecimento” e “n_maquininha”, para termos o consolidado da quantidade de maquininhas por cliente.

```
questao_tres <- transacoes %>%
  dplyr::select(estabelecimento, descricao_leitor) %>%
  unique() %>%
  dplyr::group_by(estabelecimento) %>%
  dplyr::mutate(
    n_maquininha = length(estabelecimento)
  ) %>% print()
```

```
## # A tibble: 5,340 × 3
## # Groups:   estabelecimento [4,980]
##    estabelecimento descricao_leitor           n_maquininha
##              <int> <chr>                             <int>
##  1       346777254 Maquininha PRO 2                      1
##  2       298870792 Maquininha Chip 2                     1
##  3       303139432 Maquininha PRO 2                      1
##  4       366407914 Maquininha PRO 2                      1
##  5       326302462 Maquininha PRO 2                      1
##  6       328459934 Maquininha Plus                       1
##  7       315575862 Maquininha PRO 2                      1
##  8       334005948 Maquininha Smart c/ câmera            1
##  9       285382308 Maquininha Plus                       1
## 10       327383432 Maquininha PRO 2                      3
## # ℹ 5,330 more rows
```

```
questao_tres_ <- questao_tres %>%
  dplyr::select(estabelecimento, n_maquininha) %>%
  unique() %>% print()
```

```
## # A tibble: 4,980 × 2
## # Groups:   estabelecimento [4,980]
##    estabelecimento n_maquininha
##              <int>        <int>
##  1       346777254            1
##  2       298870792            1
##  3       303139432            1
##  4       366407914            1
##  5       326302462            1
##  6       328459934            1
##  7       315575862            1
##  8       334005948            1
##  9       285382308            1
## 10       327383432            3
## # ℹ 4,970 more rows
```

Para fins ilustrativos, decido ordenar essa tabela pelo quanittativo de maquininhas.

```
questao_tres_ <- questao_tres_ %>%
  arrange(desc(n_maquininha)) %>%
  head() %>% print()
```

```
## # A tibble: 6 × 2
## # Groups:   estabelecimento [6]
##   estabelecimento n_maquininha
##             <int>        <int>
## 1       334137848            4
## 2       327383432            3
## 3       375514160            3
## 4       345823800            3
## 5       299360918            3
## 6       334145804            3
```

Qual a porcentagem de vendas com “Leitor de chip e senha” em relação a “Venda pela Maquininha”? Em número de transações e valor de faturamento total.

Para alcançar os valores desejados, precisaremos primeiro filtrar de nossa tabela, as transações que o canal de entrada foram: “Leitor de chip e senha” e “Venda pela Maquininha”. Após isso, agrupar por canal de entrada e contar a quantidade de transações para cada tipo de canal e a soma do faturamento.

```
questao_cinco <- transacoes %>%
  subset(
    descricao_canal_entrada == "Venda com leitor de chip e senha" |
    descricao_canal_entrada == "Venda pela Maquininha"
    ) %>%
  dplyr::group_by(canal_entrada) %>%
  dplyr::mutate(
    n_transacoes = length( descricao_canal_entrada ),
    sum_faturamento = sum( valor_total_transacao )
  ) %>%
  dplyr::select(descricao_canal_entrada, n_transacoes, sum_faturamento) %>%
  unique() %>% dplyr::ungroup() %>% print()
```

```
## # A tibble: 2 × 4
##   canal_entrada descricao_canal_entrada          n_transacoes sum_faturamento
##   <chr>         <chr>                                   <int>           <dbl>
## 1 ME            Venda pela Maquininha                    9911         412043.
## 2 MP            Venda com leitor de chip e senha           50          10408.
```

Levantando esses valores, basta calcular a porcentagem de vendas em relação de um canal com o outro.

```
questao_cinco_ <- questao_cinco %>%
  dplyr::mutate(

    percent_num_transacoes =
      n_transacoes[descricao_canal_entrada == "Venda com leitor de chip e senha"]/
      n_transacoes[descricao_canal_entrada == "Venda pela Maquininha"]*100,

    percent_total_vendas =
      sum_faturamento[descricao_canal_entrada == "Venda com leitor de chip e senha"]/
      sum_faturamento[descricao_canal_entrada == "Venda pela Maquininha"]*100

  ) %>%
  dplyr::select(percent_num_transacoes, percent_total_vendas) %>%
  unique() %>% print()
```

```
## # A tibble: 1 × 2
##   percent_num_transacoes percent_total_vendas
##                    <dbl>                <dbl>
## 1                  0.504                 2.53
```

Precisamos de um gráfico que demonstre o faturamento total por parcelas.

Para os gráficos, iremos utilizar o pacote *plotly* que permite a plotagem de gráficos interativos.

Primeiro precisaremos preparar os dados que serão utilizados no gráfico, para isso selecionamos apenas as coluans “quantidade_parcela” e “valor_total_tansacao”, agrupamos pela qunatidade de parcelas e realizamos a soma do valor total de transações, tendo então uma coluna com o valor de faturamento por parcela.

```
questao_seis <- transacoes %>%
  dplyr::select(quantidade_parcela, valor_total_transacao) %>%
  dplyr::group_by(quantidade_parcela) %>%
  dplyr::mutate(
    valor_por_parcela = sum(valor_total_transacao)
  ) %>%
  dplyr::select(quantidade_parcela, valor_por_parcela) %>%
  dplyr::ungroup() %>% unique() %>%
  dplyr::arrange(quantidade_parcela) %>%
  print()
```

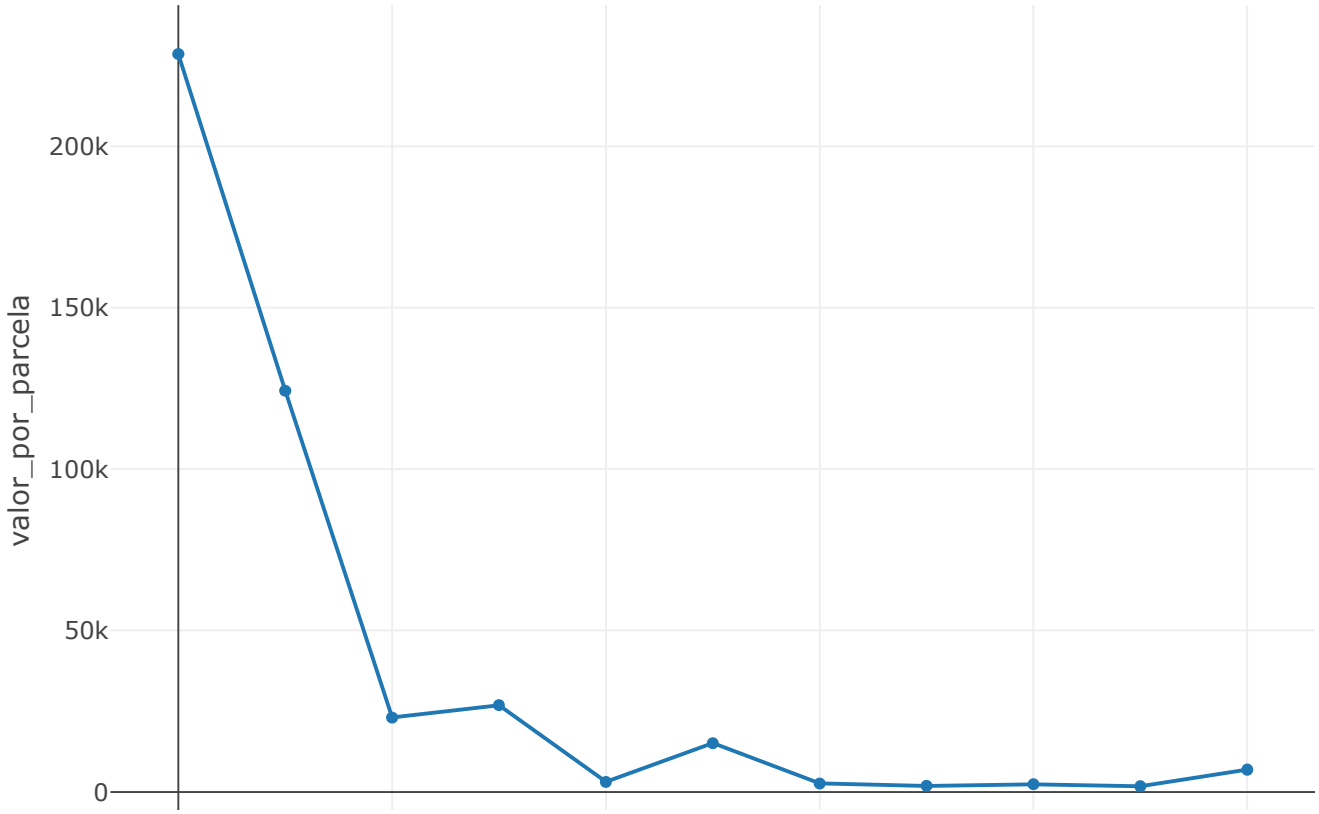
```
## # A tibble: 11 × 2
##    quantidade_parcela valor_por_parcela
##                 <int>             <dbl>
##  1                  0           228554.
##  2                  1           124274.
##  3                  2            23054.
##  4                  3            26873.
##  5                  4             3120.
##  6                  5            15108.
##  7                  6             2624.
##  8                  7             1900
##  9                  8             2400
## 10                  9             1740.
## 11                 10             6896.
```

Agora que os dados estão prontos basta montar o gráfico, que é montado de forma simples, basta indicar o dado qeu será utilizado, após isso chamar a função *plotly* e indicar os valores que preencheram os eixos, informar o tipo de gráfico e o modo.

```
plot_faturameto_parcela <- questao_seis %>%
  plotly::plot_ly(
    x = ~quantidade_parcela,
    y = ~valor_por_parcela,
    type = "scatter",
    mode = "lines+markers"
  )
```

Montado o gráfico basta chama-lo para que ele seja imprimido.

```
plot_faturameto_parcela
```



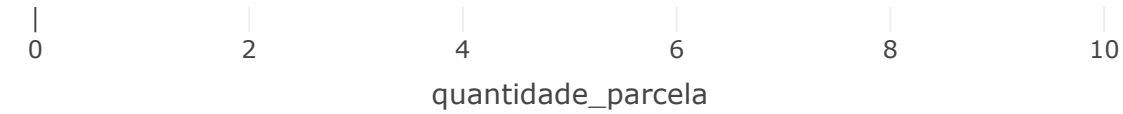


Gráfico com a razão de transações débito, crédito e parcelado.

Para essa questão, vamos verificar a quantidade de dados qeu estarão presentes em cada filtro, por exemplo: A quantidade de dados que tem como meio de pagamento o id oito (8), ou seja que são meio de pagamento do tipo cartão de débito. Levantando esses valores será possível calcular a razão de cada grupo em relação ao total de transações.

```
questao_sete <- transacoes %>%
  dplyr::mutate(
    transacoes_debito = nrow(transacoes[meio_pagamento == 8 & quantidade_parcela  < 2]),
    transacoes_credito = nrow(transacoes[meio_pagamento == 3 & quantidade_parcela  < 2]),
    transacoes_parcelado = nrow(transacoes[quantidade_parcela  > 1]),
    total_transacoes = nrow(transacoes)
  ) %>%
  select(transacoes_debito, transacoes_credito, transacoes_parcelado, total_transacoes)
%>%
  unique() %>% print()
```

```
##    transacoes_debito transacoes_credito transacoes_parcelado total_transacoes
##                <int>              <int>                <int>            <int>
## 1:              7038               2448                  159            10000
```

Agora tendo levantado esses valores, vamos montar uma tabela, utilizando a função *data.frame*, vamos montar uma tabela, com os tipos de transação e as razões de cada uma.

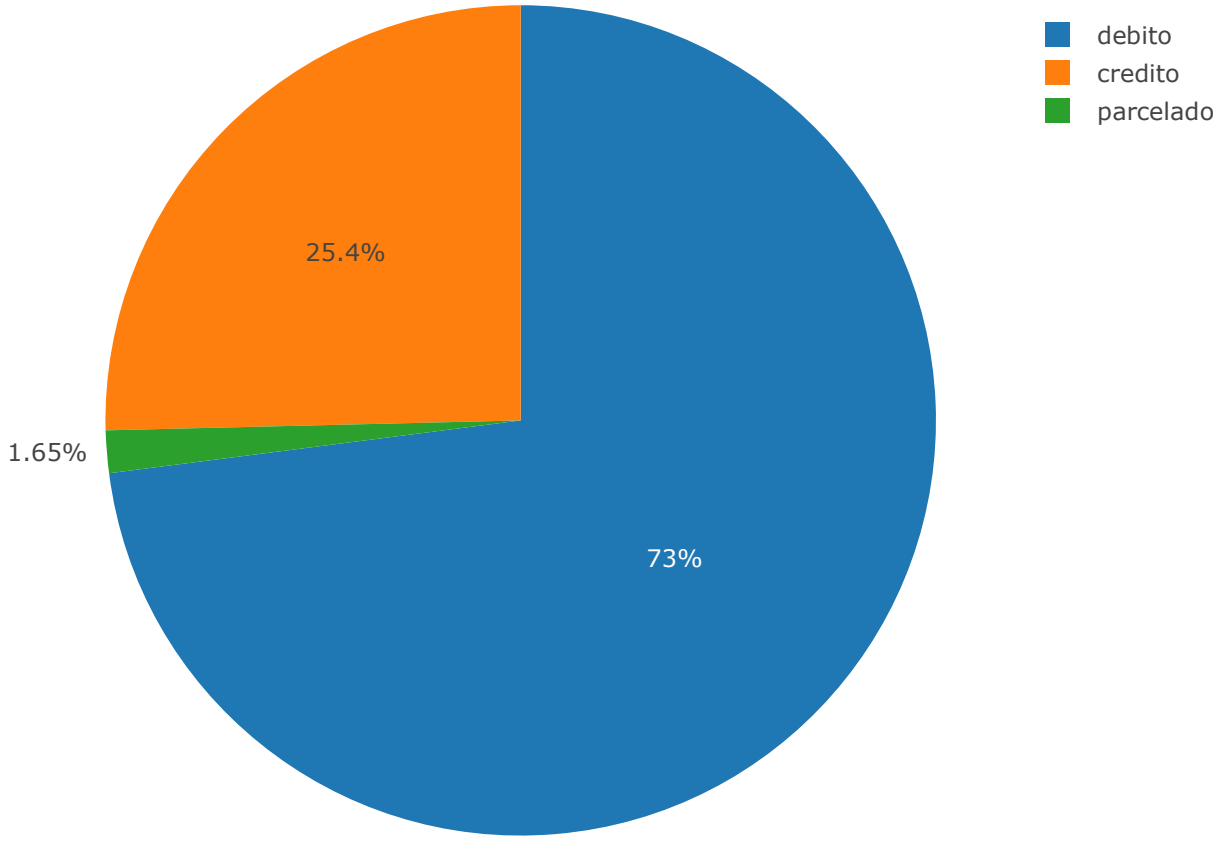
```
questao_sete_ <- data.frame(
  tipo_transacao = c(
    "debito",
    "credito",
    "parcelado"
  ),
  razao = c(
    questao_sete$transacoes_debito/questao_sete$total_transacoes,
    questao_sete$transacoes_credito/questao_sete$total_transacoes,
    questao_sete$transacoes_parcelado/questao_sete$total_transacoes
  )
) %>% print()
```

```
##   tipo_transacao  razao
## 1         debito 0.7038
## 2        credito 0.2448
## 3      parcelado 0.0159
```

Por fim, montar o gráfico:

```
plot_razao_tipo_transacao <- questao_sete_ %>%
  plotly::plot_ly(
    labels = ~tipo_transacao,
    values= ~razao,
    type = "pie"
  )

plot_razao_tipo_transacao
```



Conclusão

Tecnologia e informação, andam juntas para que as estratégias sejam eficazes, como proposta, foi realizada esta resolução das questões apresentadas nesse teste, porém não feitas em Excel e sim utilizando os recursos tecnológicos da programação, com intuito de trazer à tona essa reflexão: Quão eficaz é manter a utilização de ferramentas como planilhas eletrônicas para análises complexas e rotinerias? Sendo que a tecnologia nos oferce agilidade e praticidade nesses trabalhos.