Desafio Civitas EMD

Candidata: Stephanie Martins Pinto da Costa

Conteúdo

1 Análise Exploratória de Dados

Para iniciar a análise exploratória dos dados, primeiro foi verificado se o nome e o tipo das variáveis correspondiam com as definições apresentadas no escopo do desafio (Figura 1).

```
SELECT
  column_name,
  data_type
FROM
  rj-cetrio.desafio.INFORMATION_SCHEMA.COLUMNS
WHERE
  table_name = 'readings_2024_06';
```

Linha	column_name	data_type
1	datahora	TIMESTAMP
2	datahora_captura	TIMESTAMP
3	placa	BYTES
4	empresa	BYTES
5	tipoveiculo	BYTES
6	velocidade	INT64
7	camera_numero	BYTES
8	camera_latitude	FLOAT64
9	camera_longitude	FLOAT64

Figura 1: Resultado da verificação do nome e do tipo das variáveis.

O segundo passo envolveu a conversão das variáveis do tipo byte para base64. Essa etapa foi necessária para obter uma representação textual dos dados binários presentes nessa tabela. Como à natureza dos dados nas variáveis placa, empresa, tipo_veiculo e camera_numero indica dados oriundos de imagens. Essa conversão precisou acontecer (Figura 2).

```
-- Subconsulta para converter colunas BYTES para Base64
WITH base64_converted AS (
  SELECT
    TO_BASE64(placa) AS placa_base64,
    TO_BASE64(empresa) AS empresa_base64,
    TO_BASE64(tipoveiculo) AS tipoveiculo_base64,
    velocidade,
```

```
    TO_BASE64(camera_numero) AS camera_numero_base64,
    camera_latitude,
    camera_longitude,
    datahora,
    datahora_captura
  FROM
    rj-cetrio.desafio.readings_2024_06
)
-- Consulta principal para retornar os dados codificados em
    Base64
SELECT
  placa_base64 AS placa,
  empresa_base64 AS empresa,
  tipoveiculo_base64 AS tipoveiculo,
  velocidade,
  camera_numero_base64 AS camera_numero,
  camera_latitude,
  camera_longitude,
  datahora,
  datahora_captura
FROM
  base64_converted;
```

Linha	placa	empresa	tipoveiculo	velocidade	camera_numero
1	irWvJFronmuqeSgVkQKjo4Y=	HiVFr51Ixg==	4uACn8DT5Q==	65	2MsjWUB5+A==
2	j3QRqLJZ+t6u91LntqgkTQs=	HiVFr51Ixg==	4uACn8DT5Q==	8	2klbvZgG4w==
3	/2vaLJR59W51cTQcuNRIZa0=	HiVFr51Ixg==	4uACn8DT5Q==	66	2yebq1Sz8g==
4	3Qy5SyedW/Y21SRUoq2Oe0s=	HiVFr51Ixg==	4uACn8DT5Q==	75	YvlP8+hWMw==
5	UOH7oHmkwXfC9qA8UsyuanI=	HiVFr51Ixg==	4uACn8DT5Q==	81	aC6AdnqNLw==

Figura 2: Resultado da conversão: 5 primeiras linhas.

O terceiro passo foi verificar a existência de valores nulos ou inválidos na tabela.

```
-- Contando o nmero total de registros na tabela
SELECT COUNT(*) AS total_registros FROM
    `rj-cetrio.desafio.readings_2024_06`;

-- Verificando a quantidade de registros nulos e invlidos para
    cada coluna
SELECT
  COUNTIF(placa IS NULL OR LENGTH(placa) = 0) AS
     num_placa_invalid,
```

```
  COUNTIF(empresa IS NULL OR LENGTH(empresa) = 0) AS
      num_empresa_invalid,
  COUNTIF(tipoveiculo IS NULL OR LENGTH(tipoveiculo) = 0) AS
      num_tipoveiculo_invalid,
  COUNTIF(velocidade IS NULL OR velocidade < 0) AS
      num_velocidade_invalid,
  COUNTIF(camera_numero IS NULL OR LENGTH(camera_numero) = 0) AS
      num_camera_numero_invalid,
  COUNTIF(camera_latitude IS NULL OR camera_latitude NOT BETWEEN
      -90 AND 90) AS num_camera_latitude_invalid,
  COUNTIF(camera_longitude IS NULL OR camera_longitude NOT
      BETWEEN -180 AND 180) AS num_camera_longitude_invalid,
  COUNTIF(datahora IS NULL) AS num_datahora_invalid,
  COUNTIF(datahora_captura IS NULL) AS
      num_datahora_captura_invalid
FROM
  `rj-cetrio.desafio.readings_2024_06`;
```

Aqui foi constatado que, em um total de 36.358.536 dados, foram encontrados 1.816.325 dados nulos na coluna datahora_captura. Isso corresponde a aproximadamente 5% dos dados (Figura 3). Esses dados foram mantidos na tabela para que não fosse perdida informação útil das outras variáveis.

Linha	num_tipoveiculo_inva	num_velocidade_inva	num_camera_numer	num_camera_latitud	num_camera_longitu	num_datahora_invali	num_datahora_captu
1	0	0	0	0	0	0	1816325

Figura 3: Resultado da verificação de valores nulos ou inválidos.

O quarto passo envolveu um resumo descritivo estatístico das variáveis numéricas. As variáveis analisadas foram a velocidade, camera_latitude e camera_longitude.

```
SELECT
  COUNT(*) AS quant_total,
  AVG(velocidade) AS media_velocidade,
  MIN(velocidade) AS min_velocidade,
  MAX(velocidade) AS max_velocidade,
  STDDEV(velocidade) AS desviopadrao_velocidade,
  AVG(camera_latitude) AS media_camera_latitude,
  MIN(camera_latitude) AS min_camera_latitude,
  MAX(camera_latitude) AS max_camera_latitude,
  STDDEV(camera_latitude) AS desviopadrao_camera_latitude,
  AVG(camera_longitude) AS media_camera_longitude,
  MIN(camera_longitude) AS min_camera_longitude,
```

```
  MAX(camera_longitude) AS max_camera_longitude,
  STDDEV(camera_longitude) AS desviopadrao_camera_longitude
FROM
  rj-cetrio.desafio.readings_2024_06;
```

A análise dos dados revelou que a média de velocidade dos veículos capturados pelas câmeras é de aproximadamente 37.11 km/h, com uma velocidade mínima de 0 km/h e uma máxima de 255 km/h. O desvio padrão da velocidade é de 15.13 km/h, logo uma dispersão grande em relação à média.

Esta atenção à variação da velocidade pode sugerir a presença de veículos clonados, uma vez que essas flutuações extremas podem estar relacionadas a atividades suspeitas, explicando as velocidades elevadas encontradas na base de dados.

Já em relação às coordenadas das câmeras, a média da latitude é aproximadamente -22.75, variando de -23.86 a 0.0, com um desvio padrão de 2.01. Quanto à longitude, a média é cerca de -43.00, com mínima em -43.69 e máxima em 43.33, apresentando um desvio padrão de 3.86.

Como o desvio padrão foi pequeno e não houve flutuações discrepantes nos valores encontrados. Não foi possível muitos insights em relação a esses dados.

Assim o quinto passo foi verificar se existia alguma correlação entre velocidade e as coordenadas de latitude e longitude (Figura 4).

```
-- Verificando se existe uma correlao entre velocidade e as
    coordenadas latitude e longitude
SELECT
  CORR(velocidade, camera_latitude) AS corr_velocidade_latitude,
  CORR(velocidade, camera_longitude) AS corr_velocidade_longitude
FROM
  rj-cetrio.desafio.readings_2024_06;
```

Linha	corr_velocidade_latitude	corr_velocidade_longitude
1	0.033792642210885876	0.034843296755765078

Figura 4: Resultado da correlação.

Pelo valor encontrado não é possível dizer que existe uma correlação entre essas variáveis.

A etapa seis da análise exploratória consistiu em verificar quantas empresas estão presentes na base de dados e quantas vezes cada uma delas apareceu em relação ao total de dados (Figura 5).

```
-- Olhando as empresas
WITH empresa_counts AS (
  SELECT
    TO_BASE64(empresa) AS empresa_base64,
    COUNT(*) AS frequency
  FROM
    `rj-cetrio.desafio.readings_2024_06`
  GROUP BY
    empresa_base64
  ORDER BY
    frequency DESC
),
total_empresas AS (
  SELECT
    COUNT(*) AS total
  FROM
    `rj-cetrio.desafio.readings_2024_06`
)
SELECT
  empresa_counts.empresa_base64,
  empresa_counts.frequency,
  ROUND((empresa_counts.frequency / total_empresas.total) * 100,
     2) AS proporcao
FROM
  empresa_counts,
  total_empresas
ORDER BY
  frequency DESC;
```

Linha	empresa_base64	frequency	proporcao
1	HiVFr51Ixg==	24474488	67.31
2	CJGWe0E/pA==	8717449	23.98
3	LOAagMfz0A==	3166599	8.71

Figura 5: Resultado do estudo das empresas.

O resultado indicou um total de três empresas. A empresa identificada

por HiVFr51Ixg== é a que mais aparece, com 24.474.488 aparições, representando 67,31% dos dados.

A segunda empresa, identificada por CJGWe0E/pA==, aparece 8.717.449 vezes, representando 23,98% do total de registros. A terceira empresa, identificada por LOAagMfz0A==, corresponde a 8,71% dos registros, com 3.166.599 aparições.

Analisando as empresas, é possível observar que a empresa HiVFr51Ixg== domina a base de dados, com mais de dois terços do total de registros. Dada essa predominância, pode ser interessante verificar a quantidade de placas que esta empresa possui e se elas se repetem com frequência (Figura 6).

```
-- Olhando as empresas
WITH empresa_counts AS (
  SELECT
    TO_BASE64(empresa) AS empresa_base64,
    COUNT(*) AS frequency
  FROM
    `rj-cetrio.desafio.readings_2024_06`
  GROUP BY
    empresa_base64
  ORDER BY
    frequency DESC
),
total_empresas AS (
  SELECT
    COUNT(*) AS total
  FROM
    `rj-cetrio.desafio.readings_2024_06`
),
distinct_plates AS (
  SELECT
    TO_BASE64(empresa) AS empresa_base64,
    COUNT(DISTINCT TO_BASE64(placa)) AS placas_distintas
  FROM
    `rj-cetrio.desafio.readings_2024_06`
  GROUP BY
    empresa_base64
)

SELECT
  ec.empresa_base64,
  ec.frequency,
  ROUND((ec.frequency / te.total) * 100, 2) AS
```

```
    proporcao_registros,
  dp.placas_distintas,
  ROUND((dp.placas_distintas / te.total) * 100, 2) AS
    proporcao_placas_distintas
FROM
  empresa_counts ec
  CROSS JOIN total_empresas te
  INNER JOIN distinct_plates dp ON ec.empresa_base64 =
    dp.empresa_base64
ORDER BY
  ec.frequency DESC;
```

Linha	empresa_base64	frequency	proporcao_registros	placas_distintas	proporcao_placas_distintas
1	HiVFr51Ixg==	24474488	67.31	7260436	19.97
2	CJGWe0E/pA==	8717449	23.98	1456077	4.0
3	LOAagMfz0A==	3166599	8.71	855623	2.35

Figura 6: Resultado do estudo das empresas e suas respectivas placas.

A análise da quantidade de placas distintas de cada empresa revela que a empresa HiVFr51Ixg== possui 7.260.436 placas diferentes, correspondendo a 19,97% do total de registros. Assim, quase 20% das placas dessa empresa são únicas entre si. As placas restantes podem ser repetições de algumas das placas já contabilizadas ou podem ser placas diferentes que não foram capturadas nos dados analisados.

A empresa CJGWe0E/pA== possui 1.456.077 placas distintas, equivalente a 4% do total. Por fim, a empresa LOAagMfz0A== apresenta 855.623 placas distintas, o que representa 2,35% do total de registros.

Nessa análise, observa-se que a empresa HiVFr51Ixg== parece ter uma ampla variedade de veículos, enquanto a empresa CJGWe0E/pA== pode ter menos diversidade ou uma maior repetição de placas. Já a empresa LOAagMfz0A== mostra uma presença menor e uma variedade de placas mais limitada em comparação com as outras duas empresas.

2 Estratégia para identificar possíveis placas clonadas

Uma estratégia mais simples e direta para tentar identificar placas clonadas deriva da análise exploratória da velocidade e consiste em identificar veículos com a mesma placa detectados em locais e horários muito diferentes.

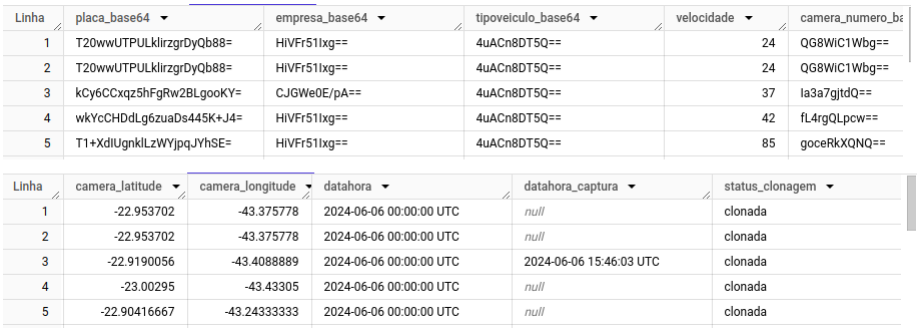
```sql
WITH base_data AS (
  SELECT
    TO_BASE64(placa) AS placa_base64,
    TO_BASE64(empresa) AS empresa_base64,
    TO_BASE64(tipoveiculo) AS tipoveiculo_base64,
    velocidade,
    TO_BASE64(camera_numero) AS camera_numero_base64,
    camera_latitude,
    camera_longitude,
    datahora,
    datahora_captura
  FROM
    `rj-cetrio.desafio.readings_2024_06`
),
-- identificar dois pontos t1 e t2
cloned_plates AS (
  SELECT
    t1.placa_base64,
    t1.datahora AS t1_datahora,
    t2.datahora AS t2_datahora,
    t1.camera_latitude AS t1_latitude,
    t1.camera_longitude AS t1_longitude,
    t2.camera_latitude AS t2_latitude,
    t2.camera_longitude AS t2_longitude,
    ST_DISTANCE(ST_GEOGPOINT(t1.camera_longitude,
        t1.camera_latitude), ST_GEOGPOINT(t2.camera_longitude,
        t2.camera_latitude)) AS distance_meters,
    ABS(TIMESTAMP_DIFF(t1.datahora, t2.datahora, SECOND)) AS
        time_diff_seconds,
    'clonada' AS status_clonagem
  FROM
    base_data t1
  JOIN
    base_data t2 ON t1.placa_base64 = t2.placa_base64
                AND t1.datahora < t2.datahora
  WHERE
  -- calcular a distncia
    ST_DISTANCE(ST_GEOGPOINT(t1.camera_longitude,
        t1.camera_latitude), ST_GEOGPOINT(t2.camera_longitude,
        t2.camera_latitude)) > 1000 -- Distncia minima em metros
        para considerar clonagem (1000 metros = 1 km)
    AND ABS(TIMESTAMP_DIFF(t1.datahora, t2.datahora, SECOND)) <
        3600 -- Tempo maximo em segundos para considerar clonagem
```

```
        (3600 segundos = 1 hora)
),

-- Calculando a proporcao de placas clonadas e nao clonadas
labeled_data AS (
  SELECT
    IF(cloned_plates.placa_base64 IS NOT NULL, 'clonada', 'nao
        clonada') AS status_clonagem,
    COUNT(*) AS count
  FROM
    cloned_plates
  GROUP BY
    status_clonagem
)

-- Incluindo os dados originais e a informacao de clonagem
SELECT
  base_data.*,
  COALESCE(cloned_plates.status_clonagem, 'nao clonada') AS
      status_clonagem
FROM
  base_data
LEFT JOIN
  cloned_plates
ON
  base_data.placa_base64 = cloned_plates.placa_base64
  AND base_data.datahora = cloned_plates.t1_datahora
ORDER BY
  base_data.datahora;
```

Com essa estratégia foi possível identificar as placas possivelmente clonadas, uma vez que só foram classificadas como clonadas as placas que indicavam estar em dois lugares ao mesmo tempo ou que apresentaram velocidades discrepantes (Figura 7).



Linha	placa_base64	empresa_base64	tipoveiculo_base64	velocidade	camera_numero_ba
1	T20wwUTPULklirzgrDyQb88=	HiVFr51Ixg==	4uACn8DT5Q==	24	QG8WiC1Wbg==
2	T20wwUTPULklirzgrDyQb88=	HiVFr51Ixg==	4uACn8DT5Q==	24	QG8WiC1Wbg==
3	kCy6CCxqz5hFgRw2BLgooKY=	CJGWe0E/pA==	4uACn8DT5Q==	37	Ia3a7gjtdQ==
4	wkYcCHDdLg6zuaDs445K+J4=	HiVFr51Ixg==	4uACn8DT5Q==	42	fL4rgQLpcw==
5	T1+XdlUgnklLzWYjpqJYhSE=	HiVFr51Ixg==	4uACn8DT5Q==	85	goceRkXQNQ==

Linha	camera_latitude	camera_longitude	datahora	datahora_captura	status_clonagem
1	-22.953702	-43.375778	2024-06-06 00:00:00 UTC	*null*	clonada
2	-22.953702	-43.375778	2024-06-06 00:00:00 UTC	*null*	clonada
3	-22.9190056	-43.4088889	2024-06-06 00:00:00 UTC	2024-06-06 15:46:03 UTC	clonada
4	-23.00295	-43.43305	2024-06-06 00:00:00 UTC	*null*	clonada
5	-22.90416667	-43.24333333	2024-06-06 00:00:00 UTC	*null*	clonada

Figura 7: Resultado da identificação das placas clonadas: as 5 primeiras linhas da classificação.