[DS_PEL] - Prova de SQL

Nome: Alessandro Melo de Oliveira

email: alessandromelo@usp.br

Como o enunciado não exigiu que as consultas fossem feitas com a sintaxe do BigQuery nem que fossem executadas no Google BigQuery, utilizei **SQLite** juntamente com o **DB Browser**.

Algumas considerações feitas sobre a implementação em SQLite:

- O SQLite não tem um type para datas, como outras bases SQL. Para os tipos envolvendo datetime, criei as colunas como TEXT.
- Tipos STRINGS também são tratados como TEXT.
- A table criada possui nome “tlc_yellow_trips_2018”.

Resolução

1. Qual foi a receita de cada tipo de pagamento no dia 15 de Março de 2018?

Pelo enunciado fornecido, há 6 tipos de pagamentos possíveis na coluna *payment_type*:

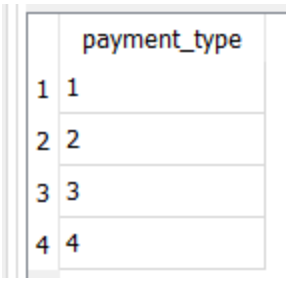
1. Credit Card
2. Cash

3. No charge
4. Dispute
5. Unknown
6. Voided trip

Entretanto, analisando os dados, há apenas 4 tipos de pagamentos, que são os tipos 1,2,3 e 4 listados acima.

```
SELECT DISTINCT payment_type
FROM tlc_yellow_trips_2018;
```

Retorno da consulta:



	payment_type
1	1
2	2
3	3
4	4

O valor total de cada corrida é dado na coluna *total_amount*.

Logo, para resolver a questão, pode-se seguir o seguinte raciocínio:

- Inicialmente, faz-se uma query selecionando apenas os dias “2018-03-15”. Isso é possível utilizando o operador LIKE, que vai checar se a string “2018-03-15” está contida dentro dos timestamps percorridos.
- Em seguida, faz-se um group by utilizando soma como função agregadora.

Em SQLite, temos:

```
SELECT payment_type, SUM(total_amount) AS "total_amount_per_payment_type"
FROM tlc_yellow_trips_2018
```

```
WHERE dropoff_datetime LIKE "2018-03-15%"
GROUP BY payment_type;
```

Cujo retorno é:

	payment_type	total_amount_per_payment_type
1	1	586.81
2	2	23.8

Logo, pode-se concluir que no dia em questão houveram pagamentos apenas em credit card e cash, cuja soma foi:

1. Credit Card: $586.81
2. Cash: $23.8

---

2. Considere que corridas de táxi válidas tenham de 1 a 5 passageiros. Qual a quantidade de corridas feitas com cada número de passageiros?

Inicialmente, faz-se uma query para retornar apenas os dados que possuem *passenger_count* com valores entre 1 e 5. Para isso utiliza-se a função BETWEEN, ou então a função IN.

A função BETWEEN é útil quando trata-se de ranges de valores inteiros contínuos, enquanto a função IN tem seu valor em casos categórios.

Provavelmente o enunciado pediu isso pois há valores nos dados contendo 0 e 6, que seriam valores não muito lógicos de se ter num taxi.

Em SQLite, temos:

```
-- Opcao 1
SELECT passenger_count, COUNT(passenger_count) AS "total_records"
FROM tlc_yellow_trips_2018
WHERE passenger_count BETWEEN 1 AND 5
GROUP BY passenger_count;
```

```
-- Opcao 2
SELECT passenger_count, COUNT(passenger_count) AS "total_records"
FROM tlc_yellow_trips_2018
WHERE passenger_count IN (1,2,3,4,5)
GROUP BY passenger_count;
```

Cujo retorno é:

	passenger_count	total_records
1	1	716
2	2	152
3	3	32
4	4	25
5	5	50

Pode-se notar que a maioria das corridas (73.4%) são feitas com apenas um único passageiro.

---

3. Considerando apenas as corridas que houveram pedágios (tolls), qual a média do valor pago em pedágios por corrida?

Felizmente os dados não possuem pedágios negativos (o que seria um erro de registro nos dados), logo é suficiente buscar apenas os *tolls_amounts* > 0;

Para calcular o valor da média, pode-se usar a função AVG.

Em SQLite, temos:

```
SELECT AVG(tolls_amount) as "average_tolls_amount"
FROM tlc_yellow_trips_2018
WHERE tolls_amount > 0;
```

Cujo retorno é:

	average_tolls_amount
1	6.7378187919463

Logo, considerando apenas as corridas que houveram pedágio, o preço médio do pedágio ofoi de $6.73.

---

4. Qual a hora que mais começaram corridas?

Essa é um pouco mais complicada. Como o formato de tempo no dados é do tipo YYYY-MM-DDTHH:mm:ss, o que precisamos fazer é buscar uma maneira de acessar o valor “HH” e ver qual é valor de hora mais recorrente.

Uma maneira de realizar isso é dividir a data com base em posições:

Y	Y	Y	Y	-	M	M	-	D	D	T	H	H	:	m	m	:	s	s
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Na tabela acima, “HH” corresponde as posições 12 e 13. Logo, é possível dar um split em cada data e buscar essas posições utilizando a função substr(pickup_datetime, 12, 2), onde 12 corresponde ao primeiro H, e 2 é o comprimento da substring procurada.

Selecionado a nova coluna contendo as horas, seleciona-se também a quantidade de vezes em que cada horário aparece, e em seguida realia-ze um group by pela hora de ínicio das corridas.

Logo, em SQLite:

```
SELECT substr(pickup_datetime, 12, 2) AS "pickup_hour", COUNT(*) AS "total_records"
FROM tlc_yellow_trips_2018
GROUP BY pickup_hour
ORDER BY total_records DESC
LIMIT 1;
```

Cujo retorno é:

	pickup_hour	total_records
1	22	70

Logo, pode-se notar que o horário que mais começam corridas são às 22:00, com 70 registros (14.61% do total de corridas).