SQL ANALÍTICO

BIG DATA

POR QUÊ (1)?

SQL é suportada nativamente por diferentes ferramentas de visualização de dados (*BI tools*, *Data Visualization Tools*)

SQL é comumente empregada em ferramentas de preparação / transformação de dados (*ETL Tools*)

POR QUÊ (2)?

Plataformas analíticas suportam a linguagem SQL (ANSI, extensões) ou linguagens inspiradas em SQL

Pré-processamento de dados em Bancos de Dados / Plataformas Big Data reduzem volume de dados e delegam custo de processamento para ambientes mais capazes

Padrão ISO/ANSI

O padrão guia as implementações da linguagem SQL para compatibilidade entre diferentes produtos

Incorpora evoluções de produtos, padronizando sua semântica para demais produtos que adotarem o padrão

Padrão ISO/ANSI

ENQUETE

Você conhece os seguintes recursos da linguagem SQL?

- *Common Table Expressions*
- *Window Functions*
- *Unbounded Arrays, Nested data*

Você conhece os seguintes recursos da linguagem SQL?

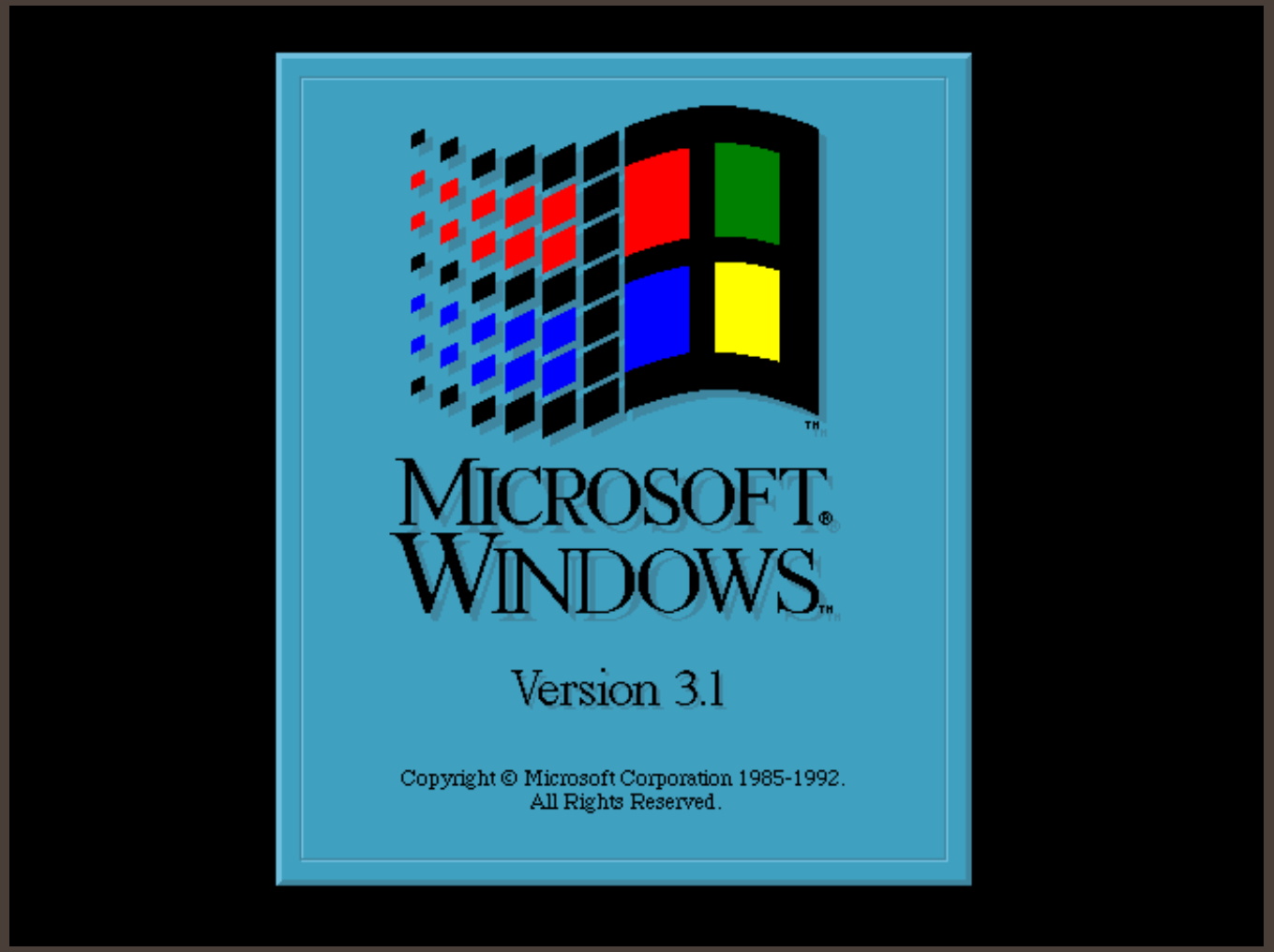
[1999] *Common Table Expressions*

[2003] *Window Functions*

[2008] *Unbounded Arrays, Nested data*

ANO DO PADRÃO

Você usa esse Windows?



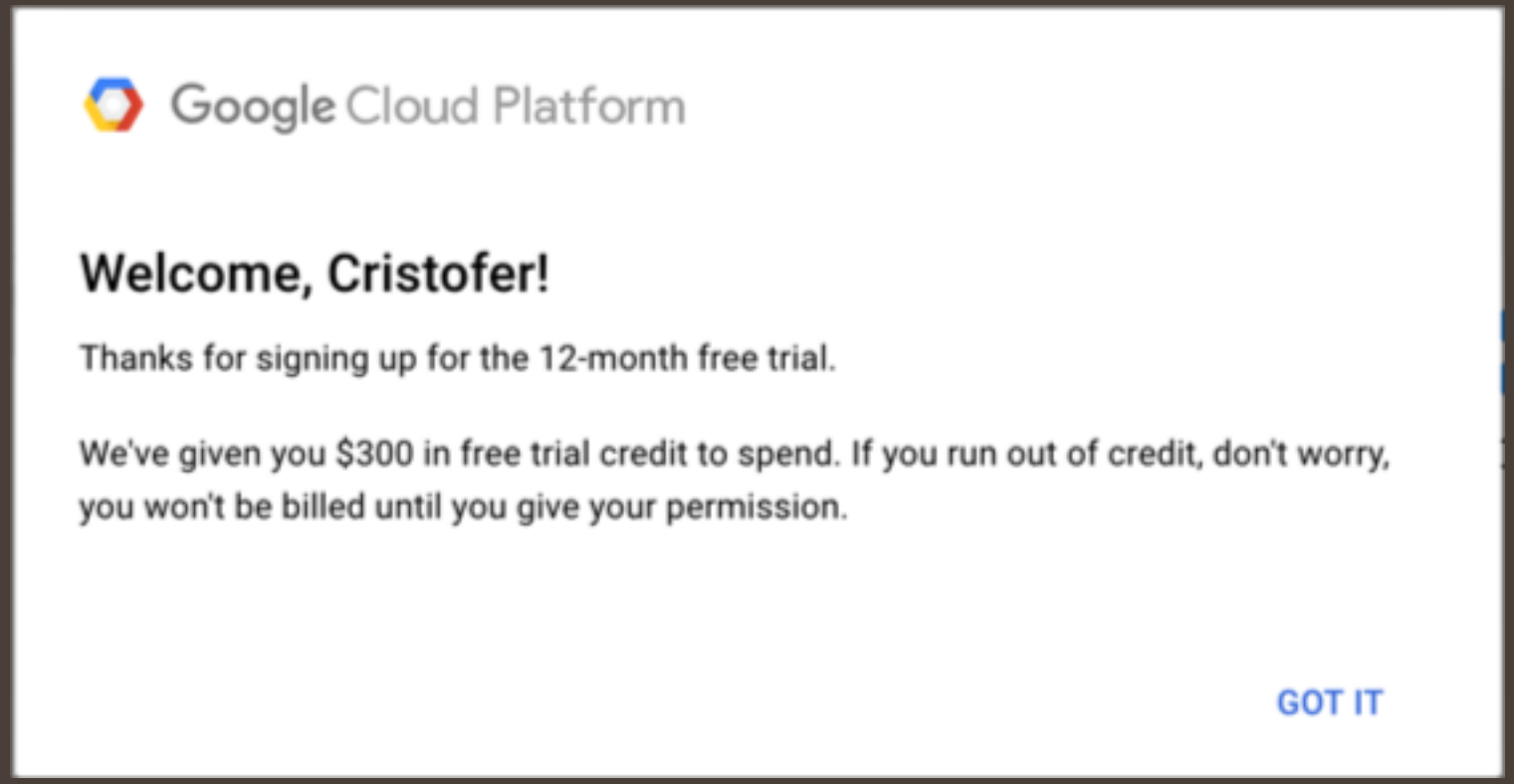
Serviço (sem servidores) de data warehouse

Baixo custo

Escalonável (escalável)

AMBIENTE
Google BigQuery

SETUP


Google Cloud Platform
Welcome, Cristofer!
Thanks for signing up for the 12-month free trial.
We've given you $300 in free trial credit to spend. If you run out of credit, don't worry, you won't be billed until you give your permission.
GOT IT

SETUP

- https://cloud.google.com/bigquery/public-data/
- criar projeto

SETUP


Error: Encountered " "WITH" "WITH "" at line 1, column 1. Was expecting: <EOF> [Try using standard SQL (https://cloud.google.com/bigquery/docs/reference/standard-sql/enabling-standard-sql)]
Destination Table
Select Table
No table selected
Write Preference
Write if empty
Append to table
Overwrite table
Results Size
Allow Large Results
Results Schema
Flatten Results
Query Caching
Use Cached Results
Query Priority
Interactive
Batch
UDF Source URIs
Edit
Maximum Bytes Billed
Project Default
SQL Dialect
Use Legacy SQL
Destination Encryption
Default
Processing Location
Unspecified
RUN QUERY
Save Query
Save View
Format Query
Hide Options
Desabilitar

SELECT

Construção select-from-where

SELECT [**ALL** | **DISTINCT**] ('*' | *column_name* ',' ...)
[**FROM** *table_name* ',' ...]
[**WHERE** *search_condition*]
[**GROUP BY** *column_name* ',' ...]
[**HAVING** *search_condition*]
[**ORDER BY** *ordering_spec* ',' ...]
[**LIMIT** *posint*]

DATASET NYC TAXI

This dataset includes trip records from all trips completed in yellow taxis in NYC since 2009.

Records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts.

The data used in the attached datasets were collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab Passenger Enhancement Program (TPEP).

DATASET NYC TAXI

vendor_id	STRING	A code indicating the TPEP provider that provided the record. 1= Creative Mobile Technologies, LLC; 2= VeriFone Inc
pickup_datetime	TIMESTAMP	The date and time when the meter was engaged.
dropoff_datetime	TIMESTAMP	The date and time when the meter was disengaged.
passenger_count	INTEGER	The number of passengers in the vehicle. This is a driver-entered value
trip_distance	FLOAT	The elapsed trip distance in miles reported by the taximeter.
pickup_longitude	FLOAT	Longitude where the meter was engaged.
pickup_latitude	FLOAT	Latitude where the meter was engaged.
rate_code	INTEGER	The final rate code in effect at the end of the trip. 1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride
store_and_fwd_flag	STRING	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka "store and forward," because the vehicle did not have a connection to the server. Y= store and forward trip N= not a store and forward trip
dropoff_longitude	FLOAT	Longitude where the meter was disengaged
dropoff_latitude	FLOAT	Latitude where the meter was disengaged.
payment_type	STRING	A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip
fare_amount	FLOAT	The time-and-distance fare calculated by the meter
extra	FLOAT	Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
mta_tax	FLOAT	$0.50 MTA tax that is automatically triggered based on the metered rate in use
tip_amount	FLOAT	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included
tolls_amount	FLOAT	Total amount of all tolls paid in trip.
imp_surcharge	FLOAT	$0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
total_amount	FLOAT	The total amount charged to passengers. Does not include cash tips
pickup_location_id	INTEGER	Describe this field…
dropoff_location_id	INTEGER	Describe this field…

```sql
SELECT passenger_count
      , round(trip_distance) as rounded_distance
      , count(*) as trips_this_year
  FROM [bigquery-public-data:new_york_taxi_trips.tlc_yellow_trips_2017]
 GROUP BY passenger_count
        , rounded_distance
HAVING trips_this_year > 1
 ORDER BY trips_this_year desc
 LIMIT 5
```

EXEMPLO

ATIVIDADE

Utilizando a construção select-from-where, determine o preço (fare_amount) médio por milha percorrida (trip_distance) para os diferentes códigos de tarifa (rate_code)

Atenção:

- Qual cuidado tomar com distâncias menores que uma milha?

Pergunta:

- A documentação de rate_code condiz com o resultado?

Ajuda com as funções:

https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators

CTE

- **Cláusula WITH**
  - contém uma ou mais subconsultas nomeadas cuja saída atua como uma tabela temporária.
- Essa saída pode ser usada como referência em instruções SELECT posteriores de qualquer cláusula ou subconsulta
- Outra função útil da cláusula WITH é dividir consultas mais complexas
- Permite criar colunas calculadas a partir de colunas de uma tabela e depois utilizar o resultado em cálculos posteriors na mesma tabela

```sql
  WITH q_trip_type AS (
          SELECT rate_code
               , trip_distance
               , fare_amount
               , IF(trip_distance < 1, "short", "long") AS trip_distance_type
            FROM `bigquery-public-data.new_york_taxi_trips.tlc_yellow_trips_2017` )
     , q_rate AS (
          SELECT rate_code
               , AVG(fare_amount / IF(trip_distance_type = "short", 1, trip_distance)) AS avg_fare_amount_per_mile
            FROM q_trip_type
           GROUP BY rate_code )
SELECT trip_distance_type
     , y.rate_code
     , count(*) AS occurrences
  FROM q_trip_type AS y
 INNER JOIN q_rate AS q
    ON y.rate_code = q.rate_code
 WHERE y.fare_amount < q.avg_fare_amount_per_mile
 GROUP BY trip_distance_type
     , y.rate_code
 ORDER BY y.rate_code, trip_distance_type;
```

EXEMPLO

ATIVIDADE

Modifique a consulta anterior substituindo a contagem de ocorrências pela fração de corridas que tiveram tarifa menor que a média calculada.

WINDOW FUNCTIONS

- As Window Functions operam em um conjunto de linhas e retornam um valor único para cada linha da consulta subjacente.
- O termo janela descreve o conjunto de linhas nas quais a função opera.
- Uma função de janela usa valores das linhas de uma janela para calcular os valores retornados.

```sql
WITH q_daily_avg_fare AS (
     SELECT AVG( fare_amount ) AS AVG_FARE
          , rate_code
          , TIMESTAMP_TRUNC( pickup_datetime, DAY ) AS PICKUP_DATE
       FROM `bigquery-public-data.new_york_taxi_trips.tlc_yellow_trips_2017`
      WHERE pickup_datetime >= '2017-01-01'
        AND rate_code <> 99
      GROUP BY PICKUP_DATE
             , rate_code )
 SELECT rate_code
      , PICKUP_DATE
      , AVG_FARE
      , AVG_FARE - LAG(AVG_FARE, 1) OVER ( PARTITION BY rate_code ORDER BY PICKUP_DATE ) AS DIFF_PREVIOUS
   FROM q_daily_avg_fare
  ORDER BY rate_code, PICKUP_DATE;
```

EXEMPLO

ATIVIDADE

- Com base na consulta anterior, modifique a cláusula WITH para calcular a soma total de tarifas (fare_amount) e então calcule a média móvel de 7 dias desta soma.
- Verifique na documentação de funções analíticas do BigQuery como definir o tamanho de uma janela (window_frame_clause)
  - https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators#analytic-functions

UNBOUNDED ARRAYS & NESTED DATA

- Versões anteriores do padrão SQL possibilitavam a criação de tipos como ARRAY & STRUCT
- No entanto, estes tipos eram predefinidos. Não existiam operações para modifica-los ou cria-los dinamicamente
- Formatos hierárquicos como XML, JSON, Avro demandaram o suporte a tipos dinâmicos

```sql
WITH rides_from_pickup_location AS (
     SELECT pickup_location_id
          , TIMESTAMP_TRUNC( pickup_datetime, DAY ) AS PICKUP_DATE
          , ARRAY_AGG(STRUCT(dropoff_location_id, fare_amount, rate_code, trip_distance) ORDER BY trip_distance ASC) AS destinations
       FROM `bigquery-public-data.new_york_taxi_trips.tlc_yellow_trips_2017`
      WHERE rate_code <> 99
        AND dropoff_location_id <> pickup_location_id
      GROUP BY pickup_location_id
          , PICKUP_DATE
)
SELECT * FROM rides_from_pickup_location
```

EXEMPLO

ATIVIDADE

- Modifique a consulta anterior para retornar somente registros dos dias 01/01/2017 e 31/12/2017
- Selecione somente os registros cujo ARRAY possua mais de 1 destino.
  - Veja nas funções de ARRAY como determinar o comprimento de um ARRAY: https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators#array-functions
- Limite o resultado em 50 registros
- Leia o arquivo JSON utilizando o sparklyr
- Converta o dataframe lido para um arquivo Avro