PROGRAMACIÓN PARA CIENCIA DE DATOS II

## 

**Jorge Andres Melo Mayorga**

Facultad de Ingeniería, Ciencia de Datos, Universidad Compensar

Actividad Practica Aplicada, Etapa de Contextualización

**Sebastián Rodriguez Muñoz**

Docente

01 de septiembre de 2024

Realiza un análisis exploratorio de los datos, aplicando los métodos, técnicas de estadística y de probabilidad para determinar tendencias, teniendo en cuenta los conceptos de las principales herramientas de manejo de software. La siguiente actividad le permitirá aplicar sus habilidades de programación en Python para abordar

problemas de Data Science.

A través de esta actividad, identificará un problema real, seleccionará los datos necesarios y realizará un análisis exploratorio inicial para obtener información valiosa.

Esta actividad se realizará individualmente, tendrá la oportunidad de aplicar lo aprendido de manera autónoma y experimentar con la programación en Python para el análisis de datos. Utilizando Jupyter se debe evidenciar el desarrollo de las siguientes actividades y contestar las preguntas, según sea correspondiente:

# PROBLEMA

*Defina un problema específico dentro del campo elegido que pueda abordarse mediante el análisis de datos. Explica el problema y por qué vale la pena investigarlo.*

## Descripción de la base de datos

Conjunto de datos públicos de comercio electrónico brasileño de pedidos realizados en Olist Store. El conjunto de datos tiene información de 100 000 pedidos de 2016 a 2018 realizados en varios mercados de Brasil. Sus características permiten ver un pedido desde múltiples dimensiones: desde el estado del pedido, el precio, el pago y el rendimiento del flete hasta la ubicación del cliente, los atributos del producto y, finalmente, las reseñas escritas por los clientes. También publicamos un conjunto de datos de geolocalización que relaciona los códigos postales brasileños con las coordenadas de latitud y longitud. Se trata de datos comerciales reales, que han sido anonimizados y las referencias a las empresas y socios en el texto de la reseña han sido sustituidas por los nombres de las grandes casas de Juego de Tronos.

## Definición del problema

Realizare la identificación de las áreas geográficas que presentan mayores demoras en las entregas de pedidos, a través del análisis espacial de los datos de geolocalización de clientes y vendedores, junto con la información sobre tiempos de entrega y costos de envío, para identificar patrones geográficos que contribuyen a retrasos en la entrega, con el fin de realizar estrategias de optimización logística, como el ajuste de rutas de envío, la apertura de nuevos centros de distribución o la colaboración con transportistas locales, para mejorar la eficiencia de la logística a la vez de reducir los tiempos de entrega.

# BASE DE DATOS DEMOGRÁFICA

¿Cómo el análisis de datos podría contribuir a comprender o resolver el problema definido? ¿Qué ideas espera obtener?

## Hipótesis, identificación inicial.

Mi análisis tiene como objetivo comprender cómo los factores geoespaciales, los tiempos de entrega y los costos de envío se relacionan con los retrasos, con el fin de optimizar la logística y mejorar la eficiencia en las entregas. Para lograrlo, me enfocaré en los siguientes aspectos:

* Identificar los tiempos de entrega y los costos asociados en diferentes regiones para detectar áreas geográficas específicas donde los retrasos son más frecuentes.
* Utilizar modelos de regresión para analizar cómo variables como la distancia, el costo de envío y los métodos de transporte afectan los tiempos de entrega. Esto me permitirá cuantificar la influencia de cada factor y priorizar las acciones de mejora.
* Examinar la variabilidad en los tiempos de entrega en función de factores como los días de la semana, las horas pico y las condiciones meteorológicas. Esta información será crucial para mejorar la planificación logística.
* Evaluar el impacto de los costos de envío en los tiempos de entrega y en la satisfacción del cliente. Esto me ayudará a determinar si es necesario ajustar las tarifas o renegociar acuerdos con proveedores logísticos.
* Segmentar a los clientes según su disposición a pagar más por envíos más rápidos. Con esta información, podré diseñar estrategias de precios diferenciales que optimicen los ingresos sin sacrificar la eficiencia operativa.

**Resultados Esperados:**

* **Mapa de Calor**: Crear un mapa de calor que visualice las áreas con más retrasos, lo que me permitirá enfocar los esfuerzos de mejora en las zonas geográficas con mayores problemas.
* **Identificación de Factores Clave**: Descubrir los principales factores que contribuyen a los retrasos en las entregas, como largas distancias, infraestructuras deficientes o costos elevados.
* **Estrategias de Optimización**: Sugerir estrategias como la apertura de nuevos centros de distribución en áreas estratégicas, el ajuste de rutas de entrega o la colaboración con transportistas locales para mejorar la eficiencia y reducir los tiempos de entrega.
* **Satisfacción del Cliente**: Mejorar la precisión en los tiempos de entrega y reducir los retrasos, lo cual no solo optimizará la operación logística, sino que también aumentará la satisfacción del cliente.

## Construcción de la base de datos

1. Se crea la base de datos a través de la Shell sql (psql)

* “CREATE DATABASE olist\_ecommerce;”
* \c olist\_ecommerce
* CREATE EXTENSION postgis;

1. Despliego dbeaver para Crear las tablas, ver código anexo
2. Importa los datos a partir de código, ver código anexo
3. Total de datos importados

|  |  |
| --- | --- |
| **table\_name** | **row\_count** |
| olist\_geolocation | 1000163 |
| olist\_order\_items | 112650 |
| olist\_order\_payments | 103886 |
| olist\_order\_customers | 99441 |
| olist\_orders | 99441 |
| temp\_order\_reviews | 98410 |
| olist\_order\_reviews | 98410 |
| olist\_products | 32951 |
| spatial\_ref\_sys | 8500 |
| olist\_sellers | 3095 |
| product\_category\_name\_translation | 71 |

# Medidas Estadísticas

Realiza un cálculo de al menos dos medidas estadísticas descriptivas clave (por ejemplo, media, mediana, desviación estándar) utilizando NumPy. ¿Qué información te proporcionan estas medidas sobre los datos?

## Conectar BD en Python

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## Medidas de tendencia central a tablas

## Tabla olist order customers

Tiene como objetivo recopilar y organizar las opiniones de los clientes sobre sus compras. Como resultado se obtiene que la mayoría de las reseñas están concentradas en las calificaciones más altas, especialmente en la calificación de 5, que representa el 58% del total. Esto sugiere que una gran proporción de clientes están muy satisfechos con su experiencia. La media es mayor que la mediana, lo que indica que los valores están sesgados hacia la derecha, es decir, hay más calificaciones altas que bajas. La mayoría de las opiniones se agrupan en los extremos, con una concentración en la calificación 5, donde la mayoría tuvo una experiencia positiva y la minoría significativa tuvo una experiencia muy negativa.

Texto, Tabla

Descripción generada automáticamente con confianza media

## Tabla olist\_order\_reviews

Utilizando una base de datos PostgreSQL y mediante la integración de coordenadas geográficas basadas en códigos postales. Esto fue posible gracias a la tabla olist\_geolocation, que contiene datos de latitud y longitud asociados a cada prefijo de código postal. Con esta información, generé un mapa que visualiza las ciudades con más de mil pedidos, permitiendo un análisis geoespacial preciso de la distribución de los clientes."

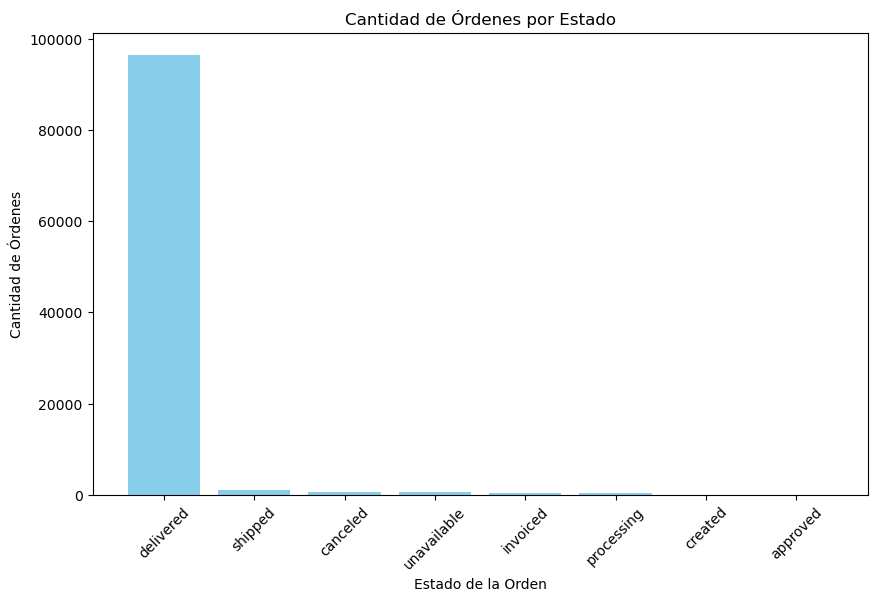
Mapa

Descripción generada automáticamente

## Tabla Olist Orders

Proporciona información el estado del pedido, la mayor parte de este se encuentra entregado.

|  |  |
| --- | --- |
| **Estado del Pedido** | **Cantidad de Órdenes** |
| entregado | 96,478 |
| enviado | 1,107 |
| cancelado | 625 |
| no disponible | 609 |
| facturado | 314 |
| en procesamiento | 301 |
| creado | 5 |
| aprobado | 2 |



## Tabla Olist Order Payments

La tabla olist\_order\_payments almacena información detallada sobre los pagos realizados para cada pedido. Cada registro en la tabla se identifica mediante payment\_id, mientras que el campo order\_id enlaza el pago específico con el pedido correspondiente. El campo payment\_sequential indica la secuencia de pagos en caso de que un pedido tenga múltiples transacciones asociadas. El tipo de pago utilizado en la transacción se registra en payment\_type, y payment\_installments captura el número de cuotas en las que se ha dividido el pago. Finalmente, el campo payment\_value almacena el valor monetario total del pago realizado.

|  |  |
| --- | --- |
| **Tipo de Pago** | **Cantidad Ordenes** |
| Tarjeta de crédito | 76,795 |
| boleto | 19,784 |
| cupón | 5,775 |
| Tarjeta de débito | 1,529 |
| No definido | 3 |

Gráfico, Gráfico de barras

Descripción generada automáticamente

**Pago Total: 16.008.872,12**

**ANEXOS 1 –   
Código PostgreSQL**

- ANALISIS DE DATOS

---Olist Store brazil

-- Crear base de datos en shell

---- CREATE DATABASE olist\_ecommerce;

---- \c olist\_ecommerce

---- CREATE EXTENSION postgis;

-- Crear Tablas

--- Tabla de clientes

**CREATE** **TABLE** olist\_order\_customers (

customer\_id **VARCHAR** **PRIMARY** **KEY**,

customer\_unique\_id **VARCHAR**,

customer\_zip\_code\_prefix **INTEGER**,

customer\_city **VARCHAR**,

customer\_state **VARCHAR**,

**location** GEOGRAPHY(POINT) -- Almacena la latitud y longitud del cliente

);

--- Tabla de geolocalización

**CREATE** **TABLE** olist\_geolocation (

geolocation\_id SERIAL **PRIMARY** **KEY**,

zip\_code\_prefix **INTEGER**,

geolocation\_lat **FLOAT**,

geolocation\_lng **FLOAT**,

geolocation\_city **VARCHAR**,

geolocation\_state **VARCHAR**,

geom GEOGRAPHY(POINT, 4326) -- Geometría espacial con sistema de referencia WGS 84

);

--- Tabla de vendedores

**CREATE** **TABLE** olist\_sellers (

seller\_id **VARCHAR** **PRIMARY** **KEY**,

seller\_zip\_code\_prefix **INTEGER**,

seller\_city **VARCHAR**,

seller\_state **VARCHAR**,

**location** GEOGRAPHY(POINT) -- Almacena la latitud y longitud del vendedor

);

--- Tabla de productos

**CREATE** **TABLE** olist\_products (

product\_id **VARCHAR** **PRIMARY** **KEY**,

product\_category\_name **VARCHAR**,

product\_name\_lenght **INTEGER**,

product\_description\_lenght **INTEGER**,

product\_photos\_qty **INTEGER**,

product\_weight\_g **INTEGER**,

product\_length\_cm **INTEGER**,

product\_height\_cm **INTEGER**,

product\_width\_cm **INTEGER**

);

--- Tabla de pedidos

**CREATE** **TABLE** olist\_orders (

order\_id **VARCHAR** **PRIMARY** **KEY**,

customer\_id **VARCHAR** **REFERENCES** olist\_order\_customers(customer\_id),

order\_status **VARCHAR**,

order\_purchase\_timestamp **TIMESTAMP**,

order\_approved\_at **TIMESTAMP**,

order\_delivered\_carrier\_date **TIMESTAMP**,

order\_delivered\_customer\_date **TIMESTAMP**,

order\_estimated\_delivery\_date **TIMESTAMP**

);

--- Tabla de items de pedido

**CREATE** **TABLE** olist\_order\_items (

order\_item\_id SERIAL **PRIMARY** **KEY**,

order\_id **VARCHAR** **REFERENCES** olist\_orders(order\_id),

product\_id **VARCHAR** **REFERENCES** olist\_products(product\_id),

seller\_id **VARCHAR** **REFERENCES** olist\_sellers(seller\_id),

shipping\_limit\_date **TIMESTAMP**,

price **FLOAT**,

freight\_value **FLOAT**

);

**ALTER** **TABLE** olist\_order\_items **DROP** **CONSTRAINT** olist\_order\_items\_pkey;

**ALTER** **TABLE** olist\_order\_items **ADD** **PRIMARY** **KEY** (order\_id, order\_item\_id, product\_id);

--- Tabla de pagos

**CREATE** **TABLE** olist\_order\_payments (

payment\_id SERIAL **PRIMARY** **KEY**,

order\_id **VARCHAR** **REFERENCES** olist\_orders(order\_id),

payment\_sequential **INTEGER**,

payment\_type **VARCHAR**,

payment\_installments **INTEGER**,

payment\_value **FLOAT**

);

--- Tabla de reseñas

**CREATE** **TABLE** olist\_order\_reviews (

review\_id **VARCHAR** **PRIMARY** **KEY**,

order\_id **VARCHAR** **REFERENCES** olist\_orders(order\_id),

review\_score **INTEGER**,

review\_comment\_title **VARCHAR**,

review\_comment\_message **TEXT**,

review\_creation\_date **TIMESTAMP**,

review\_answer\_timestamp **TIMESTAMP**

);

--- Tabla de traducciones de categorías de productos

**CREATE** **TABLE** product\_category\_name\_translation (

product\_category\_name **VARCHAR** **PRIMARY** **KEY**,

product\_category\_name\_english **VARCHAR**

);

-- Importar datos .csv con manejo para duplicados

-- 1. Importar tabla olist\_order\_customers - Registros: 99441

**copy** olist\_order\_customers(customer\_id, customer\_unique\_id, customer\_zip\_code\_prefix, customer\_city, customer\_state)

**FROM** 'D:\UCOMPENSAR\Datos\olist\_customers\_dataset.csv' **DELIMITER** ',' **CSV** **HEADER**;

--- Agregar coordenadas en la columna location a traves de la tabla temporal

-- 2. Importar tabla olist\_geolocation - Registros: 1000163

**copy** olist\_geolocation(zip\_code\_prefix, geolocation\_lat, geolocation\_lng, geolocation\_city, geolocation\_state)

**FROM** 'D:\UCOMPENSAR\Datos\olist\_geolocation\_dataset.csv' **DELIMITER** ',' **CSV** **HEADER**;

--- Agregar coordenadas

**UPDATE** public.olist\_order\_customers c

**SET** **location** = **ST\_SetSRID**(**ST\_MakePoint**(tzc.most\_frequent\_lng, tzc.most\_frequent\_lat), 4326)

**FROM** temp\_zip\_code\_coor tzc

**WHERE** c.customer\_zip\_code\_prefix = tzc.zip\_code\_prefix

**returning** \*;

**CREATE** **TEMP** **TABLE** temp\_zip\_code\_coor **AS**

**SELECT**

zip\_code\_prefix,

**MODE**() **WITHIN** **GROUP** (**ORDER** **BY** geolocation\_lat) **AS** most\_frequent\_lat,

**MODE**() **WITHIN** **GROUP** (**ORDER** **BY** geolocation\_lng) **AS** most\_frequent\_lng

**FROM**

public.olist\_geolocation

**GROUP** **BY**

zip\_code\_prefix;

--- Actualizar la columna geom

**UPDATE** public.olist\_geolocation g

**SET** geom = **ST\_SetSRID**(**ST\_MakePoint**(tzc.most\_frequent\_lng, tzc.most\_frequent\_lat), 4326)

**FROM** temp\_zip\_code\_coor tzc

**WHERE** g.zip\_code\_prefix = tzc.zip\_code\_prefix

**RETURNING** \*;

-- 3. Importar tabla olist\_sellers - Registros: 3095

**copy** olist\_sellers(seller\_id, seller\_zip\_code\_prefix, seller\_city, seller\_state)

**FROM** 'D:\UCOMPENSAR\Datos\olist\_sellers\_dataset.csv' **DELIMITER** ',' **CSV** **HEADER**;

-- 4. Importar tabla olist\_products - Registros: 32951

**copy** olist\_products(product\_id, product\_category\_name, product\_name\_lenght, product\_description\_lenght, product\_photos\_qty, product\_weight\_g, product\_length\_cm, product\_height\_cm, product\_width\_cm)

**FROM** 'D:\UCOMPENSAR\Datos\olist\_products\_dataset.csv' **DELIMITER** ',' **CSV** **HEADER**;

-- 5. Importar tabla olist\_orders - Registros: 99441

**copy** olist\_orders(order\_id, customer\_id, order\_status, order\_purchase\_timestamp, order\_approved\_at, order\_delivered\_carrier\_date, order\_delivered\_customer\_date, order\_estimated\_delivery\_date)

**FROM** 'D:\UCOMPENSAR\Datos\olist\_orders\_dataset.csv' **DELIMITER** ',' **CSV** **HEADER**;

-- 6. Importar tabla olist\_order\_items - Registros: 112650

**copy** olist\_order\_items(order\_id, order\_item\_id, product\_id, seller\_id, shipping\_limit\_date, price, freight\_value)

**FROM** 'D:\UCOMPENSAR\Datos\olist\_order\_items\_dataset.csv' **DELIMITER** ',' **CSV** **HEADER**;

-- 7. Importar tabla olist\_order\_payments - Registros: 103886

**copy** olist\_order\_payments(order\_id, payment\_sequential, payment\_type, payment\_installments, payment\_value)

**FROM** 'D:\UCOMPENSAR\Datos\olist\_order\_payments\_dataset.csv' **DELIMITER** ',' **CSV** **HEADER**;

-- 8. Importar y manejar duplicados en olist\_order\_reviews

-- Crear una tabla temporal

**CREATE** **TEMP** **TABLE** temp\_order\_reviews **AS**

**SELECT** \* **FROM** olist\_order\_reviews **WHERE** **false**;

-- Importar los datos en la tabla temporal

**copy** temp\_order\_reviews **FROM** 'D:\UCOMPENSAR\Datos\olist\_order\_reviews\_dataset.csv' **DELIMITER** ',' **CSV** **HEADER**;

-- Eliminar duplicados en la tabla temporal

**DELETE** **FROM** temp\_order\_reviews

**WHERE** ctid **NOT** **IN** (

**SELECT** **min**(ctid)

**FROM** temp\_order\_reviews

**GROUP** **BY** review\_id

);

-- Insertar los datos en la tabla final, manejando duplicados - Registros: 98410

**INSERT** **INTO** olist\_order\_reviews

**SELECT** \* **FROM** temp\_order\_reviews

**ON** **CONFLICT** (review\_id) **DO** **NOTHING**;

-- 9. Importar tabla product\_category\_name\_translation - Registros: 71

**copy** product\_category\_name\_translation(product\_category\_name, product\_category\_name\_english)

**FROM** 'D:\UCOMPENSAR\Datos\product\_category\_name\_translation.csv' **DELIMITER** ',' **CSV** **HEADER**;

---- Cargue finalizado, calidad de la importación de datos

**SELECT**

relname **AS** table\_name,

n\_live\_tup **AS** row\_count

**FROM**

pg\_stat\_user\_tables

**ORDER** **BY**

n\_live\_tup **DESC**;

--- Analisis de datos

---- Tabla olist\_order\_reviews

**select** \* **from** olist\_order\_reviews;

**SELECT**

review\_score **as** Calificacion,

**COUNT**(review\_id) **AS** Cantidad,

**ROUND**(**COUNT**(review\_id) \* 100.0 / (**SELECT** **COUNT**(\*) **FROM** olist\_order\_reviews)) **AS** Porcentaje

**FROM**

olist\_order\_reviews

**GROUP** **BY**

review\_score

**ORDER** **BY**

review\_score;

---- Tabla olist\_order\_customers, Ubicacion pedidos

**select** \* **from** olist\_order\_customers;

**SELECT**

customer\_city,

**COUNT**(customer\_id) **AS** CantidadPedidos,

"location"

**FROM**

olist\_order\_customers

**GROUP** **BY**

customer\_city

**HAVING**

**COUNT**(customer\_id) > 500

**ORDER** **BY**

CantidadPedidos **DESC**;

**SELECT**

customer\_city,

customer\_state,

**COUNT**(customer\_id) **AS** CantidadPedidos,

**ST\_AsText**(ST\_Centroid(ST\_Collect(**location**::geometry))) **AS** centroid\_location

**FROM**

public.olist\_order\_customers

**WHERE**

**location** **IS** **NOT** **NULL**

**GROUP** **BY**

customer\_city, customer\_state

**HAVING**

**COUNT**(customer\_id) > 1000

**ORDER** **BY**

CantidadPedidos **DESC**;

--- Tabla olist\_orders

**select** \* **from** olist\_orders;

**SELECT**

order\_status,

**COUNT**(order\_id) **AS** CantidadOrdenes

**FROM**

olist\_orders

**GROUP** **BY**

order\_status

**ORDER** **BY**

CantidadOrdenes **DESC**;

--- Tabla olist\_orders\_payments

**SELECT** **distinct**

payment\_type **AS** TipoPago,

**count**(order\_id) **AS** Cantidad

**FROM**

olist\_order\_payments

**GROUP** **BY**

payment\_type

**ORDER** **BY**

Cantidad **DESC**;

**SELECT**

**sum** (payment\_value) **AS** PagoTotal

**FROM**

olist\_order\_payments;

**ANEXOS 2 –   
Código Python**