📚 Capstone Dataset Guide and Data Dictionary

The dataset contains real e-commerce data from 2016-2018. It represents over 100,000 orders placed across multiple marketplaces in Brazil. This guide will help you understand the structure and relationships between tables.

📦 DATASET FILES

Core Files

1. `orders_dataset.csv`

What it contains: One row per order with status and timestamps

Column	Type	Description	Example
`order_id`	string	Unique order identifier (PRIMARY KEY)	`e481f51cbdc54678b7cc49136f2d6af7`
`customer_id`	string	Unique customer identifier (FOREIGN KEY)	`9ef432eb6251297304e76186b10a928d`
`order_status`	string	Order status (delivered, shipped, canceled, etc.)	`delivered`
`order_purchase_timestamp`	datetime	When customer placed order	`2017-10-02 10:56:33`
`order_approved_at`	datetime	When payment was approved	`2017-10-02 11:07:15`
`order_delivered_carrier_date`	datetime	When order was handed to carrier	`2017-10-04 19:55:00`
`order_delivered_customer_date`	datetime	When customer received order	`2017-10-10 21:25:13`
`order_estimated_delivery_date`	datetime	Estimated delivery date	`2017-10-18 00:00:00`

Key Notes:

- Contains ~99,441 orders
- Some timestamps may be null (orders not yet delivered)

---

2. `order_items_dataset.csv`

What it contains: Product details for each item in an order (one order can have multiple items)

Column	Type	Description	Example
`order_id`	string	Links to orders table (FOREIGN KEY)	`e481f51cbdc54678b7cc49136f2d6af7`
`order_item_id`	int	Sequential number of item within order	`1`
`product_id`	string	Product identifier	`b3af7e4c31c3b8b8a5f0c4c1e4f0c4c1`
`seller_id`	string	Seller identifier	`c1e4f0c4c1e4f0c4c1e4f0c4c1e4f0c4`
`shipping_limit_date`	datetime	Seller shipping limit	`2017-10-09 03:10:00`
`price`	float	Item price (in Brazilian Reals R$)	`29.99`
`freight_value`	float	Shipping cost	`8.72`

Key Notes:

- Contains ~112,650 items (more than orders because one order can have multiple items)
- To get total order value, you need to sum `price + freight_value` per order
- Alternatively, use the payments table for total

---

3. `order_payments_dataset.csv`

What it contains: Payment information (one order can have multiple payment methods)

Column	Type	Description	Example
`order_id`	string	Links to orders table (FOREIGN KEY)	`e481f51cbdc54678b7cc49136f2d6af7`
`payment_sequential`	int	Sequential number (for split payments)	`1`
`payment_type`	string	Payment method (credit_card, boleto, voucher, debit_card)	`credit_card`
`payment_installments`	int	Number of installments	`3`
`payment_value`	float	Payment amount in R$	`99.33`

Key Notes:

- **IMPORTANT:** Some orders have multiple payment methods (e.g., part credit card, part voucher)
- To get total order value, **group by order_id and SUM payment_value**
- Contains ~103,886 payment records

---

4. `customers_dataset.csv`

What it contains: Customer location information (OPTIONAL - for geographic analysis)

Column	Type	Description	Example
`customer_id`	string	Unique customer identifier (PRIMARY KEY)	`9ef432eb6251297304e76186b10a928d`
`customer_unique_id`	string	Unique ID across different orders	`f409e6cdcf1c7e3a6c9f9f9f9f9f9f9f`
`customer_zip_code_prefix`	int	First 5 digits of zip code	`3149`
`customer_city`	string	City name	`São Paulo`
`customer_state`	string	State abbreviation (2 letters)	`SP`

Key Notes:

- Contains ~99,441 customers
- Useful for geographic analysis (bonus points!)
- One customer can place multiple orders

Optional Files (For Additional Analysis)

5. `products_dataset.csv`

What it contains: Product catalog information

Column	Type	Description
`product_id`	string	Unique product identifier
`product_category_name`	string	Category in Portuguese (e.g., "cama_mesa_banho")
`product_name_length`	int	Number of characters in product name
`product_description_length`	int	Number of characters in description
`product_photos_qty`	int	Number of product photos
`product_weight_g`	int	Product weight in grams
`product_length_cm`	int	Product length
`product_height_cm`	int	Product height
`product_width_cm`	int	Product width

Use case: Analyze which product categories Champions buy vs. Lost customers

---

6. `sellers_dataset.csv`

What it contains: Seller location information

Column	Type	Description
`seller_id`	string	Unique seller identifier
`seller_zip_code_prefix`	int	First 5 digits of seller zip
`seller_city`	string	Seller city
`seller_state`	string	Seller state

Use case: Analyze seller performance or shipping patterns

---

7. `order_reviews_dataset.csv`

What it contains: Customer reviews and ratings

Column	Type	Description
`review_id`	string	Unique review identifier
`order_id`	string	Links to orders table
`review_score`	int	Rating from 1-5 stars
`review_comment_title`	string	Review title
`review_comment_message`	string	Review text (in Portuguese)
`review_creation_date`	datetime	When review was created
`review_answer_timestamp`	datetime	When seller responded

Use case: Identify unhappy customers (low review scores) for targeted retention campaigns

---

8. `geolocation_dataset.csv`

What it contains: Detailed geolocation data (zip codes to coordinates)

Column	Type	Description
`geolocation_zip_code_prefix`	int	Zip code
`geolocation_lat`	float	Latitude
`geolocation_lng`	float	Longitude
`geolocation_city`	string	City
`geolocation_state`	string	State

Use case: Create geographic heatmaps

TABLE RELATIONSHIPS

Here's how the tables connect:

```
olist_customers_dataset
    └── customer_id (PK)
            ↓
    olist_orders_dataset
        └── order_id (PK)
                ↓
        ┌───────┴───────┐
        ↓               ↓
olist_order_items   olist_order_payments
olist_order_reviews
        ↓
olist_products_dataset
olist_sellers_dataset
```

Note/s:

1. `olist_orders_dataset` → Filter to delivered orders, get customer_id and order_purchase_timestamp
2. `olist_order_payments_dataset` → Sum payment_value per order to get TotalSpent
3. Merge these two tables on `order_id`
4. Group by `customer_id`

🔍 DATA QUALITY ISSUES TO EXPECT

1. **Missing Values**

- `order_approved_at` and delivery dates may be null for non-delivered orders
- Some products have missing category names
- Some customers don't have reviews

2. **Duplicates**

- Generally clean, but always check!
- Some orders may appear multiple times if they have multiple payment methods

3. **Data Types**

- Dates are stored as strings → Convert with `pd.to_datetime()`
- IDs are hashed strings (anonymized)

4. **Language**

- Product categories are in Portuguese
- Reviews are in Portuguese (but you don't need to translate for RFM)

5. **Outliers**

- Some orders have very high or very low values
- Some shipping costs are unusual
- Check for negative values (refunds/errors)

💡 TIPS

1. **Start Simple**
   Don't try to merge all 8 tables! Focus on orders + payments first.
2. **Check Your Merges**
   After merging, always verify:

```python
print(f"Orders before merge: {len(df_orders)}")
print(f"Rows after merge: {len(df_merged)}")
print(f"Unique orders after merge: {df_merged['order_id'].nunique()}")
```

3. **Handle Multiple Payments**
   Remember: One order can have multiple payment rows!

```python
# CORRECT way to get total per order:
order_totals = df_payments.groupby('order_id')['payment_value'].sum()

# INCORRECT - Will double-count orders with multiple payments:
df_merged['payment_value'].sum()
```

4. **Filter Early**
   Filter to delivered orders BEFORE merging to reduce dataset size:

```python
df_orders_clean = df_orders[df_orders['order_status'] == 'delivered']
```

5. **Validate Your RFM**
   Sanity checks:

- Recency should be positive (days >= 0)
- Frequency should be at least 1 (everyone has at least one order)
- Monetary should be positive (no negative spend for delivered orders)
