In [1]:

```
import numpy as np
import pandas as pd
from mysql.connector import Error
import mysql.connector
from sqlalchemy import create_engine
from urllib.parse import quote_plus
```

DATA MODELS

CUSTOMER_DATASET

In [2]:

```
customers_dataset = pd.read_csv("C:/Users/hp/Desktop/data/dataset/olist_customers_dataset.CSV")
```

In [3]:

```
customers_dataset.head()
```

Out[3]:

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_stat
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	S
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	S
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	S
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	mogi das cruzes	S
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	S

GEOLOCATION_DATASET

In [4]:

```
geolocation_dataset = pd.read_csv("C:/Users/hp/Desktop/data/dataset/olist_geolocation_dataset.csv")
```

In [5]:

```
geolocation_dataset.head()
```

Out[5]:

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
0	1037	-23.545621	-46.639292	sao paulo	SP
1	1046	-23.546081	-46.644820	sao paulo	SP
2	1046	-23.546129	-46.642951	sao paulo	SP
3	1041	-23.544392	-46.639499	sao paulo	SP
4	1035	-23.541578	-46.641607	sao paulo	SP

In [6]:

```
geolocation_dataset.shape
```

Out[6]:

```
(1000163, 5)
```

ITEM_DATASET

In [7]:

```
items_dataset = pd.read_csv("C:/Users/hp/Desktop/data/dataset/olist_order_items_dataset.csv")
```

In [8]:

```
items_dataset.head()
```

Out[8]:

	order_id	order_item_id	product_id	seller_id	ship
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	2018
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	2018
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	2017

In [9]:

```
items_dataset.shape
```

Out[9]:

```
(112650, 7)
```

In [ ]:

PAYMENT_DATASET

In [10]:
```
payments_dataset= pd.read_csv("C:/Users/hp/Desktop/data/dataset/olist_order_payments_dataset.csv")
```

In [11]:
```
payments_dataset.head()
```

Out[11]:

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45

In [12]:
```
payments_dataset.shape
```

Out[12]:
```
(103886, 5)
```

REVIEWS_DATASET

In [13]:
```
reviews_dataset= pd.read_csv("C:/Users/hp/Desktop/data/dataset/olist_order_reviews_dataset.csv")
```

In [14]:
```
reviews_dataset.head()
```

Out[14]:

	review_id	order_id	review_score	review_comment_title	review_comment_m
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	NaN	
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	NaN	
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	NaN	
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	NaN	Recebi bem antes c est
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	NaN	Parabéns lojas l adorei comprar

In [15]:
```
reviews_dataset.shape
```

Out[15]:
```
(99224, 7)
```

ORDERS_DATASET

In [16]:
```
orders_dataset= pd.read_csv("C:/Users/hp/Desktop/data/dataset/olist_orders_dataset.csv")
```

In [17]:
```
orders_dataset.head()
```

Out[17]:

	order_id	customer_id	order_status	order_purchase_timestamp	order_approvec
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-1C 11:07
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07 03:24
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	2018-08 08:55
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11 19:45
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02 22:20

In [18]:
```
orders_dataset.shape
```

Out[18]:
```
(99441, 8)
```

PRODUCT_DATASET

In [19]:
```
products_dataset= pd.read_csv("C:/Users/hp/Desktop/data/dataset/olist_products_dataset.csv")
```

In [20]:
```
products_dataset.head()
```

Out[20]:

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_q
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	287.0	1
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	276.0	1
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	250.0	1
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	261.0	1
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	402.0	4

In [21]:
```
products_dataset.shape
```

Out[21]:
```
(32951, 9)
```

SELLER_DATASET

In [22]:
```
sellers_dataset= pd.read_csv("C:/Users/hp/Desktop/data/dataset/olist_sellers_dataset.csv")
```

In [23]:
```
sellers_dataset.head()
```

Out[23]:

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP
1	d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	SP
2	ce3ad9de960102d0677a81f5d0bb7b2d	20031	rio de janeiro	RJ
3	c0f3eea2e14555b6faeea3dd58c1b1c3	4195	sao paulo	SP
4	51a04a8a6bdcb23deccc82b0b80742cf	12914	braganca paulista	SP

In [24]:
```
sellers_dataset.shape
```

Out[24]:
```
(3095, 4)
```

NAME_TRANSLATION DATASET

In [25]:
```
name_translation= pd.read_csv("C:/Users/hp/Desktop/data/dataset/product_category_name_translation.csv")
```

In [26]:
```
name_translation.head()
```

Out[26]:

	product_category_name	product_category_name_english
0	beleza_saude	health_beauty
1	informatica_acessorios	computers_accessories
2	automotivo	auto
3	cama_mesa_banho	bed_bath_table
4	moveis_decoracao	furniture_decor

In [27]:
```
name_translation.shape
```

Out[27]:
```
(71, 2)
```

DIMENSION MODEL

In [28]:
```
'''
mycur = conn.cursor()
password = '@db23'
encoded_password = quote_plus(password)
engine = create_engine(f'mysql+mysqlconnector://root:{encoded_password}@localhost:3306/e_commerce_pro')
'''
```

Out[28]:
```
"\nmycur = conn.cursor()\npassword = '@db23'\nencoded_password = quote_plus(password)\nengine = create_engine(f
'mysql+mysqlconnector://root:{encoded_password}@localhost:3306/e_commerce_pro')\n"
```

In [29]:
```
#mycur = conn.cursor()
```

In [30]:
```
pip install pymysql
```

```
Requirement already satisfied: pymysql in c:\users\hp\miniconda3\envs\envprop1\lib\site-packages (1.1.1)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
import pandas as pd
```

```
from sqlalchemy import create_engine
password = '@db23'
encoded_password = quote_plus(password)
engine = create_engine(f'mysql+mysqlconnector://root:{encoded_password}@localhost:3306/e_commerce_pro')
customers_df = pd.read_sql_table('customers_dataset', con=engine)
geolocation_df = pd.read_sql_table('geolocation_dataset', con=engine)
sellers_df = pd.read_sql_table('sellers_dataset', con=engine)
merged_df1 = pd.merge(customers_df, geolocation_df, how='inner',left_on='customer_zip_code_prefix', right_on='ge
final_merged_df = pd.merge(merged_df1, sellers_df, how='inner',left_on='geolocation_zip_code_prefix', right_on=
print(final_merged_df.head())
```

FACT TABLE

In [ ]:
```
fact_orders_info = pd.DataFrame()
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
fact_orders_info['order_id']=orders_dataset['order_id']
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
fact_orders_info  = fact_orders_info.merge(payments_dataset,on='order_id',how='left')
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
fact_orders_info=fact_orders_info.merge(items_dataset,on='order_id',how='left')
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
fact_orders_info = fact_orders_info.drop(
    columns=["review_comment_title", "review_comment_message", "review_creation_date", "review_answer_timestamp
    errors='ignore'
)
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
fact_orders_info=fact_orders_info.merge(orders_dataset,on='order_id',how='left')
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
fact_orders_info = fact_orders_info.drop(
    columns=["order_status", "order_purchase_timestamp", "order_approved_at", "order_delivered_carrier_date","o
    errors='ignore'
)
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
fact_orders_info=fact_orders_info.merge(reviews_dataset,on='order_id',how='left')
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
fact_orders_info = fact_orders_info.drop(
    columns=["review_comment_title", "review_comment_message", "review_creation_date", "review_answer_timestamp
    errors='ignore'
)
```

In [ ]:
```
fact_orders_info.head()
```

fact_orders_info DATA CLEANING

In [ ]:
```
fact_orders_info.isnull().sum()
```

In [ ]:
```
fact_orders_info.dropna(inplace=True)  # Drops rows with any missing value
```

In [ ]:
```
fact_orders_info.head()
```

In [ ]:
```
nan_count = fact_orders_info.isna().sum().sum()
```

In [ ]:
```
nan_count
```

In [ ]:
```
file_path = 'C:/Users/hp/Desktop/e_commerce_project/fact_orders_info.csv'
fact_orders_info.to_csv(file_path, index=False)
```

dim_geolocation DATA CLEANING

```
In [ ]: dim_geolocation1 = pd.DataFrame()
```

```
In [ ]: dim_geolocation1 = geolocation_dataset.copy(deep=True)
```

```
In [ ]: dim_geolocation1.head()
```

```
In [ ]: dim_geolocation1.isnull().sum()
```

```
In [ ]: dim_geolocation1.dropna(inplace=True)  # Drops rows with any missing value
```

```
In [ ]: dim_geolocation1.head()
```

```
In [ ]: nan_count = dim_geolocation1.isna().sum().sum()
```

```
In [ ]: nan_count
```

```
In [ ]: file_path = 'C:/Users/hp/Desktop/e_commerce_project/dim_geolocation1.csv'
        dim_geolocation1.to_csv(file_path, index=False)
```

DIM_CUSTOMERS DATA CLEANING

```
In [ ]: dim_customers = customers_dataset.copy(deep=True)
```

```
In [ ]: dim_customers.head()
```

```
In [ ]: dim_customers.isnull().sum()
```

```
In [ ]: dim_customers.dropna(inplace=True)  # Drops rows with any missing value
```

```
In [ ]: dim_customers.head()
```

```
In [ ]: dim_customers.shape
```

```
In [ ]: nan_count = dim_customers.isna().sum().sum()
```

```
In [ ]: nan_count
```

TO SAVE CSV FILE

```
In [ ]: file_path = 'C:/Users/hp/Desktop/e_commerce_project/dim_geolocation1.csv'
        dim_geolocation1.to_csv(file_path, index=False)
```

DIM_SELLER_DATASET

```
In [ ]: dim_seller = sellers_dataset.copy(deep=True)
```

```
In [ ]: dim_customers.isnull().sum()
```

```
In [ ]: dim_seller.dropna(inplace=True)  # Drops rows with any missing value
```

```
In [ ]: dim_seller.head()
```

```
In [ ]: dim_seller.shape
```

```
In [ ]: nan_count = dim_seller.isna().sum().sum()
```

```
In [ ]: nan_count
```

```
In [ ]: file_path = 'C:/Users/hp/Desktop/e_commerce_project/dim_seller.csv'
        dim_seller.to_csv(file_path, index=False)
```

DIM_PAYMENT_DATASET

```
In [ ]: dim_payments = payments_dataset.copy(deep=True)
```

```
In [ ]: dim_payments.head()
```

```
In [ ]: dim_payments.isnull().sum()
In [ ]: dim_payments.dropna(inplace=True)  # Drops rows with any missing value
```

```
In [ ]: dim_payments.head()
```

```
In [ ]: dim_payments.shape
```

```
In [ ]: nan_count = dim_payments.isna().sum().sum()
```

```
In [ ]: nan_count
```

```
In [ ]: file_path = 'C:/Users/hp/Desktop/e_commerce_project/dim_payments.csv'
        dim_payments.to_csv(file_path, index=False)
```

DIM_REVIEWS_DATASET

```
In [ ]: dim_reviews = reviews_dataset.copy(deep=True)
```

```
In [ ]: dim_reviews.head()
```

```
In [ ]: dim_reviews.isnull().sum()
```

```
In [ ]: dim_reviews.dropna(inplace=True)  # Drops rows with any missing value
```

```
In [ ]: dim_reviews.head()
```

```
In [ ]: dim_reviews.shape
```

```
In [ ]: nan_count = dim_reviews.isna().sum().sum()
```

```
In [ ]: nan_count
```

```
In [ ]: file_path = 'C:/Users/hp/Desktop/e_commerce_project/dim_reviews.csv'
        dim_reviews.to_csv(file_path, index=False)
```

DIM_PRODUCTS_DATASETS

```
In [151… dim_products = products_dataset.copy(deep=True)
```

```
In [152… dim_products.head()
```

Out[152…

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_q
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	287.0	1
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	276.0	1
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	250.0	1
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	261.0	1
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	402.0	4

```
In [153… dim_products.isnull().sum()
```

```
Out[153… product_id                      0
         product_category_name         610
         product_name_lenght           610
         product_description_lenght    610
         product_photos_qty            610
         product_weight_g                2
         product_length_cm               2
         product_height_cm               2
         product_width_cm                2
         dtype: int64
```

```
In [154… dim_products.dropna(inplace=True)  # Drops rows with any missing value
```

```
In [155… dim_products.head()
```

Out[155…

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_q
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	287.0	1
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	276.0	1
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	250.0	1
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	261.0	1
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	402.0	4

In [156…

```
nan_count = dim_products.isna().sum().sum()
```

In [157…

```
nan_count
```

Out[157…

```
0
```

In [158…

```
file_path = 'C:/Users/hp/Desktop/e_commerce_project/dim_products.csv'
dim_products.to_csv(file_path, index=False)
```

DIM_ORDERS_DATASET

In [159…

```
dim_orders = orders_dataset.copy(deep=True)
```

In [160…

```
dim_orders
```

Out[160…

	order_id	customer_id	order_status	order_purchase_timestamp	order_app
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	20
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	20
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	20
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	20
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	20
...	...	...	...	...	
99436	9c5dedf39a927c1b2549525ed64a053c	39bd1228ee8140590ac3aca26f2dfe00	delivered	2017-03-09 09:54:05	20
99437	63943bddc261676b46f01ca7ac2f7bd8	1fca14ff2861355f6e5f14306ff977a7	delivered	2018-02-06 12:58:58	20
99438	83c1379a015df1e13d02aae0204711ab	1aa71eb042121263aafbe80c1b562c9c	delivered	2017-08-27 14:46:43	20
99439	11c177c8e97725db2631073c19f07b62	b331b74b18dc79bcdf6532d51e1637c1	delivered	2018-01-08 21:28:27	20
99440	66dea50a8b16d9b4dee7af250b4be1a5	edb027a75a1449115f6b43211ae02a24	delivered	2018-03-08 20:57:30	20

99441 rows × 8 columns

In [161…

```
orders_dataset.head()
```

Out[161…

	order_id	customer_id	order_status	order_purchase_timestamp	order_approvec
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-10 11:07
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07 03:24
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	2018-08 08:55
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11 19:45
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02 22:20

In [162…

```
dim_orders_date = pd.DataFrame()
```

In [163…
```
dim_orders_date = orders_dataset.copy(deep=True)
```

In [164…
```
dim_orders_date.head()
```

Out[164…

	order_id	customer_id	order_status	order_purchase_timestamp	order_approve
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-1 11:07
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07 03:24
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	2018-08 08:55
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11 19:45
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02 22:20

In [165…
```
date_diam = pd.DataFrame()
```

In [166…
```
dates = pd.concat([
    dim_orders_date['order_purchase_timestamp'].dropna(),
    dim_orders_date['order_approved_at'].dropna(),
    dim_orders_date['order_delivered_carrier_date'].dropna(),
    dim_orders_date['order_delivered_customer_date'].dropna(),
    dim_orders_date['order_estimated_delivery_date'].dropna(),
]).drop_duplicates().reset_index(drop=True)
```

In [167…
```
dates = pd.to_datetime(dates)
```

In [168…
```
dates
```

Out[168…
```
0        2017-10-02 10:56:33
1        2018-07-24 20:41:37
2        2018-08-08 08:38:49
3        2017-11-18 19:28:06
4        2018-02-13 21:18:39
                 ...
363324   2016-12-23 00:00:00
363325   2017-01-11 00:00:00
363326   2016-10-25 00:00:00
363327   2018-07-10 00:00:00
363328   2016-10-27 00:00:00
Length: 363329, dtype: datetime64[ns]
```

In [169…
```
date_diam = pd.DataFrame({
    'date' : dates,
    'date_id' : range(1,len(dates) + 1)
})
```

In [170…
```
print(date_diam.columns)
```

```
Index(['date', 'date_id'], dtype='object')
```

In [171…
```
date_diam.head()
```

Out[171…

	date	date_id
0	2017-10-02 10:56:33	1
1	2018-07-24 20:41:37	2
2	2018-08-08 08:38:49	3
3	2017-11-18 19:28:06	4
4	2018-02-13 21:18:39	5

In [172…
```
date_diam['year'] = date_diam['date'].dt.year
```

In [173…
```
date_diam.head()
```

Out[173…

	date	date_id	year
0	2017-10-02 10:56:33	1	2017
1	2018-07-24 20:41:37	2	2018
2	2018-08-08 08:38:49	3	2018
3	2017-11-18 19:28:06	4	2017
4	2018-02-13 21:18:39	5	2018

In [174…

```
date_diam['month'] = date_diam['date'].dt.month
```

In [175…

```
date_diam.head()
```

Out[175…

	date	date_id	year	month
0	2017-10-02 10:56:33	1	2017	10
1	2018-07-24 20:41:37	2	2018	7
2	2018-08-08 08:38:49	3	2018	8
3	2017-11-18 19:28:06	4	2017	11
4	2018-02-13 21:18:39	5	2018	2

In [176…

```
date_diam['quarter'] = date_diam['date'].dt.quarter
```

In [177…

```
date_diam.head()
```

Out[177…

	date	date_id	year	month	quarter
0	2017-10-02 10:56:33	1	2017	10	4
1	2018-07-24 20:41:37	2	2018	7	3
2	2018-08-08 08:38:49	3	2018	8	3
3	2017-11-18 19:28:06	4	2017	11	4
4	2018-02-13 21:18:39	5	2018	2	1

In [178…

```
date_diam['day'] = date_diam['date'].dt.day
```

In [179…

```
date_diam.head()
```

Out[179…

	date	date_id	year	month	quarter	day
0	2017-10-02 10:56:33	1	2017	10	4	2
1	2018-07-24 20:41:37	2	2018	7	3	24
2	2018-08-08 08:38:49	3	2018	8	3	8
3	2017-11-18 19:28:06	4	2017	11	4	18
4	2018-02-13 21:18:39	5	2018	2	1	13

In [180…

```
date_diam['day_of_week'] = date_diam['date'].dt.day_of_week
```

In [181…

```
date_diam.head()
```

Out[181…

	date	date_id	year	month	quarter	day	day_of_week
0	2017-10-02 10:56:33	1	2017	10	4	2	0
1	2018-07-24 20:41:37	2	2018	7	3	24	1
2	2018-08-08 08:38:49	3	2018	8	3	8	2
3	2017-11-18 19:28:06	4	2017	11	4	18	5
4	2018-02-13 21:18:39	5	2018	2	1	13	1

In [182…

```
date_diam.dropna(inplace=True)  # Drops rows with any missing value
```

In [183…

```
date_diam.head()
```

Out[183…

	date	date_id	year	month	quarter	day	day_of_week
0	2017-10-02 10:56:33	1	2017	10	4	2	0
1	2018-07-24 20:41:37	2	2018	7	3	24	1
2	2018-08-08 08:38:49	3	2018	8	3	8	2
3	2017-11-18 19:28:06	4	2017	11	4	18	5
4	2018-02-13 21:18:39	5	2018	2	1	13	1

In [184…

```
nan_count = date_diam.isna().sum().sum()
```

In [185…

```
nan_count
```

Out[185…

```
0
```

In [186…

```
file_path = 'C:/Users/hp/Desktop/e_commerce_project/date_diam.csv'
date_diam.to_csv(file_path, index=False)
```

In [ ]:

In [187…

```
dim_orders_datessF = dim_orders.copy(deep=True)
```

In [188…

```
dim_orders_datessF.head()
```

Out[188…

	order_id	customer_id	order_status	order_purchase_timestamp	order_approvec
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-10 11:07
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07 03:24
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	2018-08 08:55
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11 19:45
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02 22:20

In [189…

```
dim_orders_datessF.dtypes
```

Out[189…

```
order_id                         object
customer_id                      object
order_status                     object
order_purchase_timestamp         object
order_approved_at                object
order_delivered_carrier_date     object
order_delivered_customer_date    object
order_estimated_delivery_date    object
dtype: object
```

In [190…

```
# Update original DataFrame columns to datetime
dim_orders_datessF['order_purchase_timestamp'] = pd.to_datetime(dim_orders_datessF['order_purchase_timestamp'],
dim_orders_datessF['order_approved_at'] = pd.to_datetime(dim_orders_datessF['order_approved_at'], errors='coerce
dim_orders_datessF['order_delivered_carrier_date'] = pd.to_datetime(dim_orders_datessF['order_delivered_carrier_
dim_orders_datessF['order_delivered_customer_date'] = pd.to_datetime(dim_orders_datessF['order_delivered_custome
dim_orders_datessF['order_estimated_delivery_date'] = pd.to_datetime(dim_orders_datessF['order_estimated_delive
```

In [191…

```
dim_orders_datessF.dtypes
```

Out[191…

```
order_id                                 object
customer_id                              object
order_status                             object
order_purchase_timestamp         datetime64[ns]
order_approved_at                datetime64[ns]
order_delivered_carrier_date     datetime64[ns]
order_delivered_customer_date    datetime64[ns]
order_estimated_delivery_date    datetime64[ns]
dtype: object
```

In [192…

```
dim_orders_datessF['order_purchase_timestamp_key'] = pd.to_datetime(dim_orders_datessF['order_purchase_timestam
dim_orders_datessF = pd.merge(
    dim_orders_datessF,
    date_diam[['date', 'date_id']],
    left_on='order_purchase_timestamp_key',
    right_on='date',
    how='left'
).drop(columns=['date', 'order_purchase_timestamp_key'])
```

```
dim_orders_datessF['order_purchase_timestamp_key'] = dim_orders_datessF['date_id']
dim_orders_datessF = dim_orders_datessF.drop(columns='date_id')
dim_orders_datessF['order_approved_at_key'] = pd.to_datetime(dim_orders_datessF['order_approved_at'])
dim_orders_datessF = pd.merge(
    dim_orders_datessF,
    date_diam[['date', 'date_id']],
    left_on='order_approved_at_key',
    right_on='date',
    how='left'
).drop(columns=['date', 'order_approved_at_key'])
dim_orders_datessF['order_approved_at_key'] = dim_orders_datessF['date_id']
dim_orders_datessF = dim_orders_datessF.drop(columns='date_id') t


dim_orders_datessF['order_delivered_carrier_date_key'] = pd.to_datetime(dim_orders_datessF['order_delivered_car
dim_orders_datessF = pd.merge(
    dim_orders_datessF,
    date_diam[['date', 'date_id']],
    left_on='order_delivered_carrier_date_key',
    right_on='date',
    how='left'
).drop(columns=['date', 'order_delivered_carrier_date_key'])
dim_orders_datessF['order_delivered_carrier_date_key'] = dim_orders_datessF['date_id']
dim_orders_datessF = dim_orders_datessF.drop(columns='date_id')
dim_orders_datessF['order_delivered_customer_date_key'] = pd.to_datetime(dim_orders_datessF['order_delivered_cu
dim_orders_datessF = pd.merge(
    dim_orders_datessF,
    date_diam[['date', 'date_id']],
    left_on='order_delivered_customer_date_key',
    right_on='date',
    how='left'
).drop(columns=['date', 'order_delivered_customer_date_key'])
dim_orders_datessF['order_delivered_customer_date_key'] = dim_orders_datessF['date_id']
dim_orders_datessF = dim_orders_datessF.drop(columns='date_id')
dim_orders_datessF['order_estimated_delivery_date_key'] = pd.to_datetime(dim_orders_datessF['order_estimated_de
dim_orders_datessF = pd.merge(
    dim_orders_datessF,
    date_diam[['date', 'date_id']],
    left_on='order_estimated_delivery_date_key',
    right_on='date',
    how='left'
).drop(columns=['date', 'order_estimated_delivery_date_key'])
dim_orders_datessF['order_estimated_delivery_date_key'] = dim_orders_datessF['date_id']
dim_orders_datessF = dim_orders_datessF.drop(columns='date_id')
```

In [193…

```
dim_orders_datessF.head()
```

Out[193…

	order_id	customer_id	order_status	order_purchase_timestamp	order_approvec
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-10 11:07
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07 03:24
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	2018-08 08:55
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11 19:45
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02 22:20

In [194…

```
dim_orders_datessF.dtypes
```

Out[194…

```
order_id                                  object
customer_id                               object
order_status                              object
order_purchase_timestamp          datetime64[ns]
order_approved_at                 datetime64[ns]
order_delivered_carrier_date      datetime64[ns]
order_delivered_customer_date     datetime64[ns]
order_estimated_delivery_date     datetime64[ns]
order_purchase_timestamp_key               int64
order_approved_at_key                    float64
order_delivered_carrier_date_key         float64
order_delivered_customer_date_key        float64
order_estimated_delivery_date_key          int64
dtype: object
```

In [195…

```
dim_orders_datessF.drop(columns=[
    'order_purchase_timestamp',
    'order_approved_at',
    'order_delivered_carrier_date',
    'combined_dates',
    'order_delivered_customer_date',
    'order_estimated_delivery_date',
    'order_purchase_timestamp_date_id',
    'order_approved_at_date_id',
    'order_delivered_carrier_date_id',
    'order_delivered_customer_date_id',
    'order_estimated_delivery_date_id',
    'date_id_x',
    'date_id_y'
], errors='ignore', inplace=True)
```

In [196…

```
dim_orders_datessF.dtypes
```

Out[196…

```
order_id                               object
customer_id                            object
order_status                           object
order_purchase_timestamp_key            int64
order_approved_at_key                 float64
order_delivered_carrier_date_key      float64
order_delivered_customer_date_key     float64
order_estimated_delivery_date_key       int64
dtype: object
```

In [197…

```
dim_orders_datessF['order_purchase_timestamp_key'] = dim_orders_datessF['order_purchase_timestamp_key'].fillna(
dim_orders_datessF['order_approved_at_key'] = dim_orders_datessF['order_approved_at_key'].fillna(-1).astype(int
dim_orders_datessF['order_delivered_carrier_date_key'] = dim_orders_datessF['order_delivered_carrier_date_key']
dim_orders_datessF['order_delivered_customer_date_key'] = dim_orders_datessF['order_delivered_customer_date_key
dim_orders_datessF['order_estimated_delivery_date_key'] = dim_orders_datessF['order_estimated_delivery_date_key
```

In [198…

```
dim_orders_datessF.dropna(inplace=True)  # Drops rows with any missing value
```

In [199…

```
dim_orders_datessF.head()
```

Out[199…

	order_id	customer_id	order_status	order_purchase_timestamp_key	order_appr
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	1	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	3	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	4	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	5	

In [200…

```
dim_orders_datessF.dtypes
```

Out[200…

```
order_id                             object
customer_id                          object
order_status                         object
order_purchase_timestamp_key          int32
order_approved_at_key                 int32
order_delivered_carrier_date_key      int32
order_delivered_customer_date_key     int32
order_estimated_delivery_date_key     int32
dtype: object
```

In [206…

```
nan_count = dim_orders_datessF.isna().sum().sum()
```

In [207…

```
nan_count
```

Out[207…

```
0
```

In [208…

```
file_path = 'C:/Users/hp/Desktop/e_commerce_project/dim_orders_datessF.csv'
dim_orders_datessF.to_csv(file_path, index=False)
```

In [209…

```
dim_orders_datessF.head()
```

Out[209…

	order_id	customer_id	order_status	order_purchase_timestamp_key	order_appr
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	1	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	3	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	4	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	5	

In [210…

```
dim_orders_date.dtypes
```

Out[210…

```
order_id                         object
customer_id                      object
order_status                     object
order_purchase_timestamp         object
order_approved_at                object
order_delivered_carrier_date     object
order_delivered_customer_date    object
order_estimated_delivery_date    object
dtype: object
```

In [211…

```
fact_orders_info.head()
```

Out[211…

	order_id	payment_sequential	payment_type	payment_installments	payment_value	order_item_id	
0	e481f51cbdc54678b7cc49136f2d6af7	1.0	credit_card	1.0	18.12	1.0	872
1	e481f51cbdc54678b7cc49136f2d6af7	3.0	voucher	1.0	2.00	1.0	872
2	e481f51cbdc54678b7cc49136f2d6af7	2.0	voucher	1.0	18.59	1.0	872
3	53cdb2fc8bc7dce0b6741e2150273451	1.0	boleto	1.0	141.46	1.0	59
4	47770eb9100c2d0c44946d9cf07ec65d	1.0	credit_card	3.0	179.12	1.0	aa4

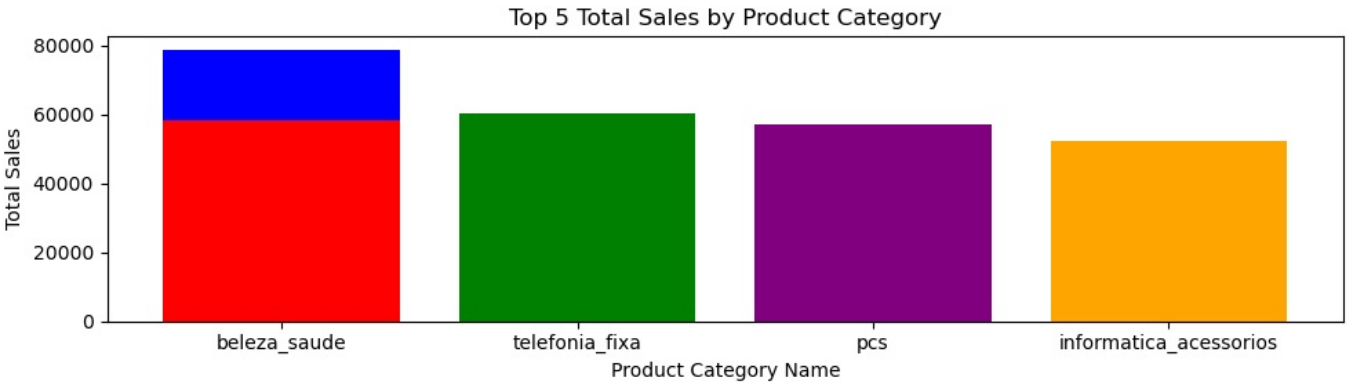
DATA VUSUALIZATION

line plot

In [212…

```
import matplotlib.pyplot as plt
import os  # Add this import
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
total_sales_by_product = fact_orders_info.groupby('product_id')['total_sales'].sum().reset_index()
print(dim_products.columns)  # Ensure 'product_category_name' is a valid column
total_sales_by_product = pd.merge(total_sales_by_product,
                                  dim_products[['product_id', 'product_category_name']],
                                  on='product_id',
                                  how='left')
total_sales_by_product['product_category_name'] = total_sales_by_product['product_category_name'].fillna('Unknow
total_sales_by_product['product_category_name'] = total_sales_by_product['product_category_name'].astype(str)
top_5_sales = total_sales_by_product.sort_values(by='total_sales', ascending=False).head(5)
colors = ['blue', 'green', 'red', 'purple', 'orange']
plt.figure(figsize=(10, 3))
plt.bar(top_5_sales['product_category_name'], top_5_sales['total_sales'], color=colors)
plt.title('Top 5 Total Sales by Product Category')
plt.ylabel('Total Sales')
plt.xlabel('Product Category Name')
plt.tight_layout()
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "top_5_sales.png")
plt.savefig(desktop_path)
plt.show()
```

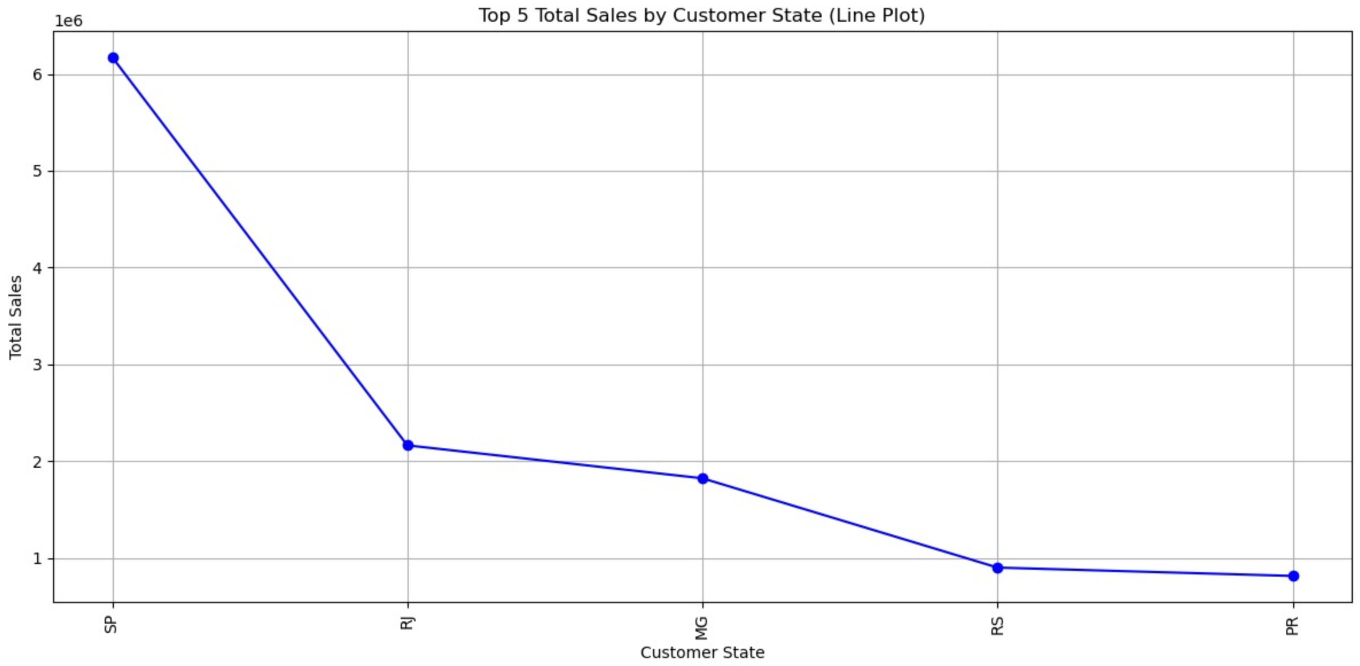
```
Index(['product_id', 'product_category_name', 'product_name_lenght',
       'product_description_lenght', 'product_photos_qty', 'product_weight_g',
       'product_length_cm', 'product_height_cm', 'product_width_cm'],
      dtype='object')
```

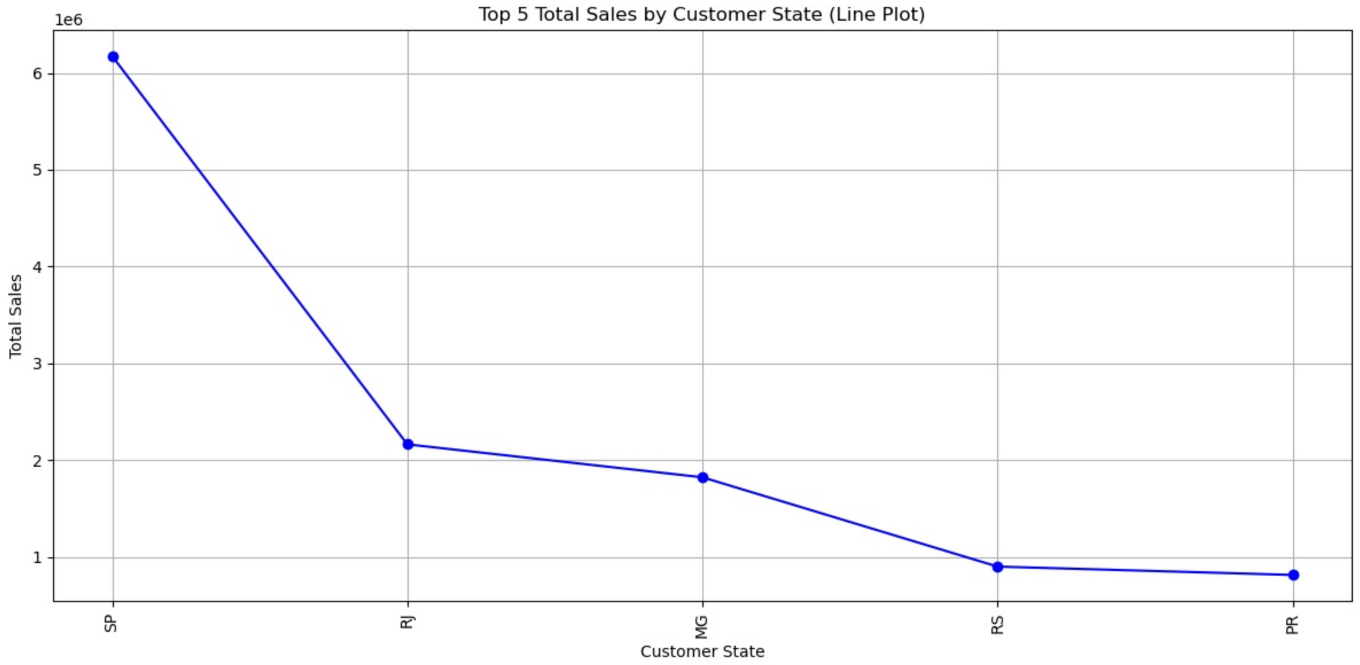


In [ ]:

In [213…

```
import pandas as pd
import matplotlib.pyplot as plt
import os
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
merged_data = pd.merge(
    fact_orders_info,
    dim_customers[['customer_id', 'customer_state']],
    on='customer_id',
    how='left',
    suffixes=('', '_dup')
)
for col in merged_data.columns:
    if '_dup' in col:
        merged_data.drop(col, axis=1, inplace=True)
total_sales_by_state = merged_data.groupby('customer_state')['total_sales'].sum().reset_index()
top_5_states = total_sales_by_state.sort_values(by='total_sales', ascending=False).head(5)
plt.figure(figsize=(12, 6))
plt.plot(top_5_states['customer_state'], top_5_states['total_sales'], marker='o', color='blue')
plt.xticks(rotation=90)
plt.title('Top 5 Total Sales by Customer State (Line Plot)')
plt.ylabel('Total Sales')
plt.xlabel('Customer State')
plt.grid(True)
plt.tight_layout()
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "top_5_sales_by_state.png")
plt.savefig(desktop_path)
plt.show()
```



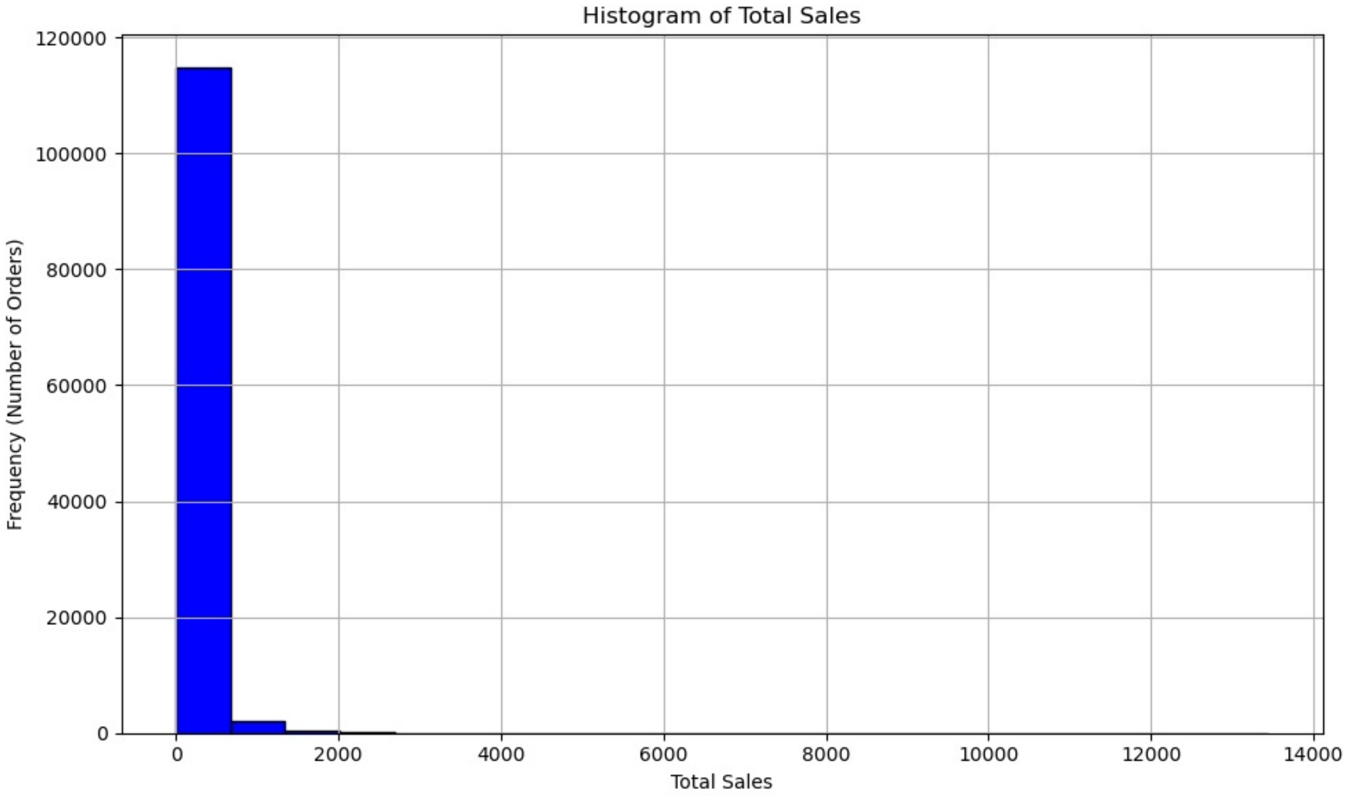


In [ ]:

In [214…

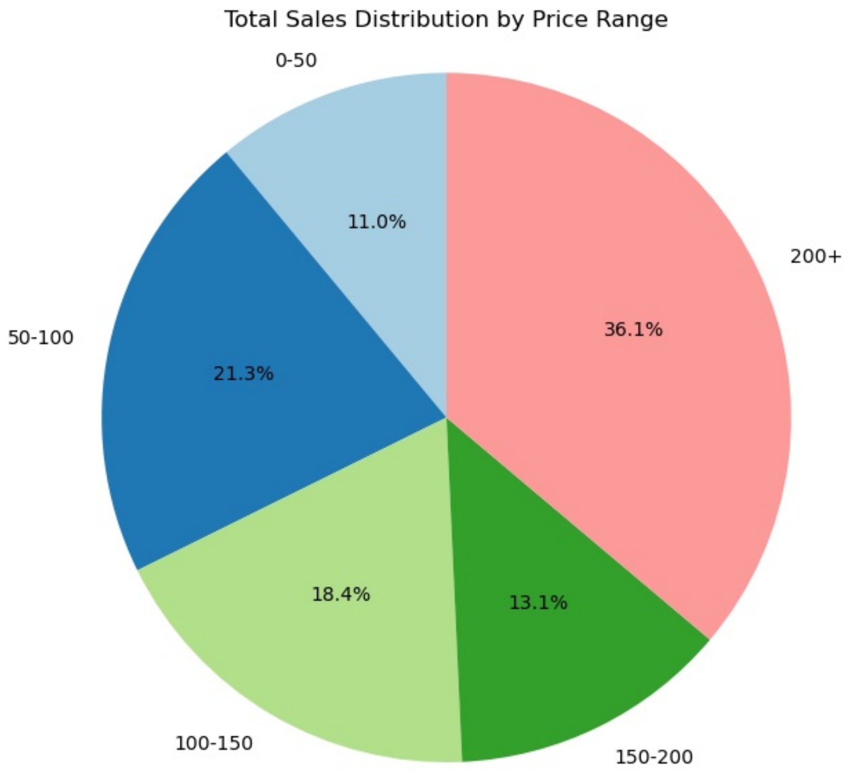
```
import pandas as pd
import matplotlib.pyplot as plt
import os
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
total_sales_data = fact_orders_info['total_sales']
```

```
plt.figure(figsize=(10, 6))
plt.hist(total_sales_data, bins=20, color='blue', edgecolor='black')
plt.title('Histogram of Total Sales')
plt.xlabel('Total Sales')
plt.ylabel('Frequency (Number of Orders)')
plt.grid(True)
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "total_sales_histogram.png")
plt.savefig(desktop_path)  # Save the histogram to the desktop
plt.tight_layout()
plt.show()
```



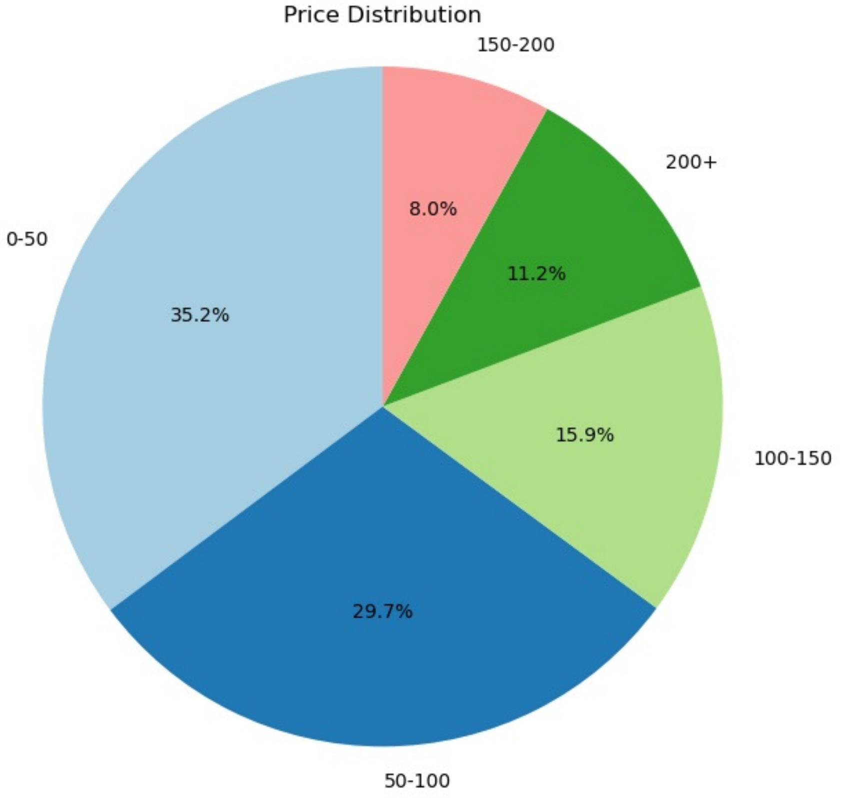
In [215…

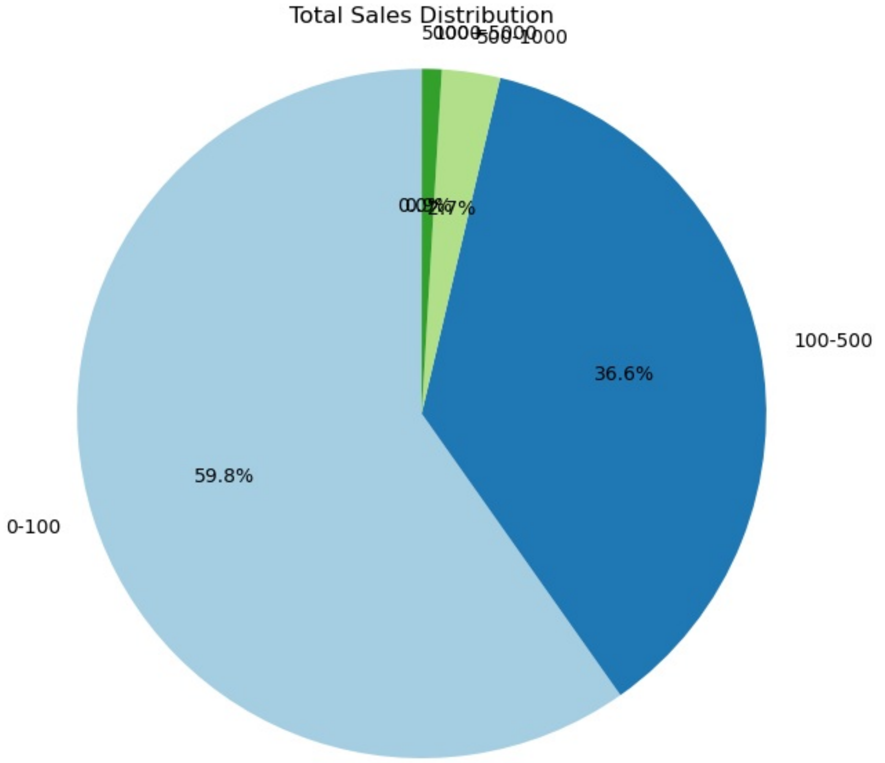
```
import pandas as pd
import matplotlib.pyplot as plt
import os
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']
bins = [0, 50, 100, 150, 200, 1000]
labels = ['0-50', '50-100', '100-150', '150-200', '200+']
fact_orders_info['price_range'] = pd.cut(fact_orders_info['price'], bins=bins, labels=labels, include_lowest=Tru
sales_by_price_range = fact_orders_info.groupby('price_range', observed=False)['total_sales'].sum()
plt.figure(figsize=(10, 6))
plt.pie(sales_by_price_range, labels=sales_by_price_range.index, autopct='%1.1f%%', startangle=90, colors=plt.c
plt.title('Total Sales Distribution by Price Range')
plt.axis('equal')
plt.tight_layout()
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "total_sales_by_price_range.png")
plt.savefig(desktop_path)
plt.show()
```



In [216…

```python
import pandas as pd
import matplotlib.pyplot as plt
import os
def plot_pie_chart(data, column, title, bins=None, labels=None, save_path=None):
    if bins and labels:
        data['binned'] = pd.cut(data[column], bins=bins, labels=labels, include_lowest=True)
        data_to_plot = data['binned'].value_counts()
    else:
        data_to_plot = data[column].value_counts()
    plt.figure(figsize=(10, 6))
    plt.pie(data_to_plot, labels=data_to_plot.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.col
    plt.title(title)
    plt.axis('equal')
    plt.tight_layout()
    if save_path:
        plt.savefig(save_path)

    plt.show()
desktop_path = os.path.expanduser("~") + "/Desktop"
if 'product_category_name' in fact_orders_info.columns:
    plot_pie_chart(
        fact_orders_info,
        'product_category_name',
        'Total Orders by Product Category',
        save_path=os.path.join(desktop_path, "total_orders_by_product_category.png")
    )
bins = [0, 50, 100, 150, 200, 1000]
labels = ['0-50', '50-100', '100-150', '150-200', '200+']
if 'price' in fact_orders_info.columns:
    plot_pie_chart(
        fact_orders_info,
        'price',
        'Price Distribution',
        bins=bins,
        labels=labels,
        save_path=os.path.join(desktop_path, "price_distribution.png")
    )
bins_sales = [0, 100, 500, 1000, 5000, 10000]
labels_sales = ['0-100', '100-500', '500-1000', '1000-5000', '5000+']
if 'total_sales' in fact_orders_info.columns:
    plot_pie_chart(
        fact_orders_info,
        'total_sales',
        'Total Sales Distribution',
        bins=bins_sales,
        labels=labels_sales,
        save_path=os.path.join(desktop_path, "total_sales_distribution.png")
    )
```

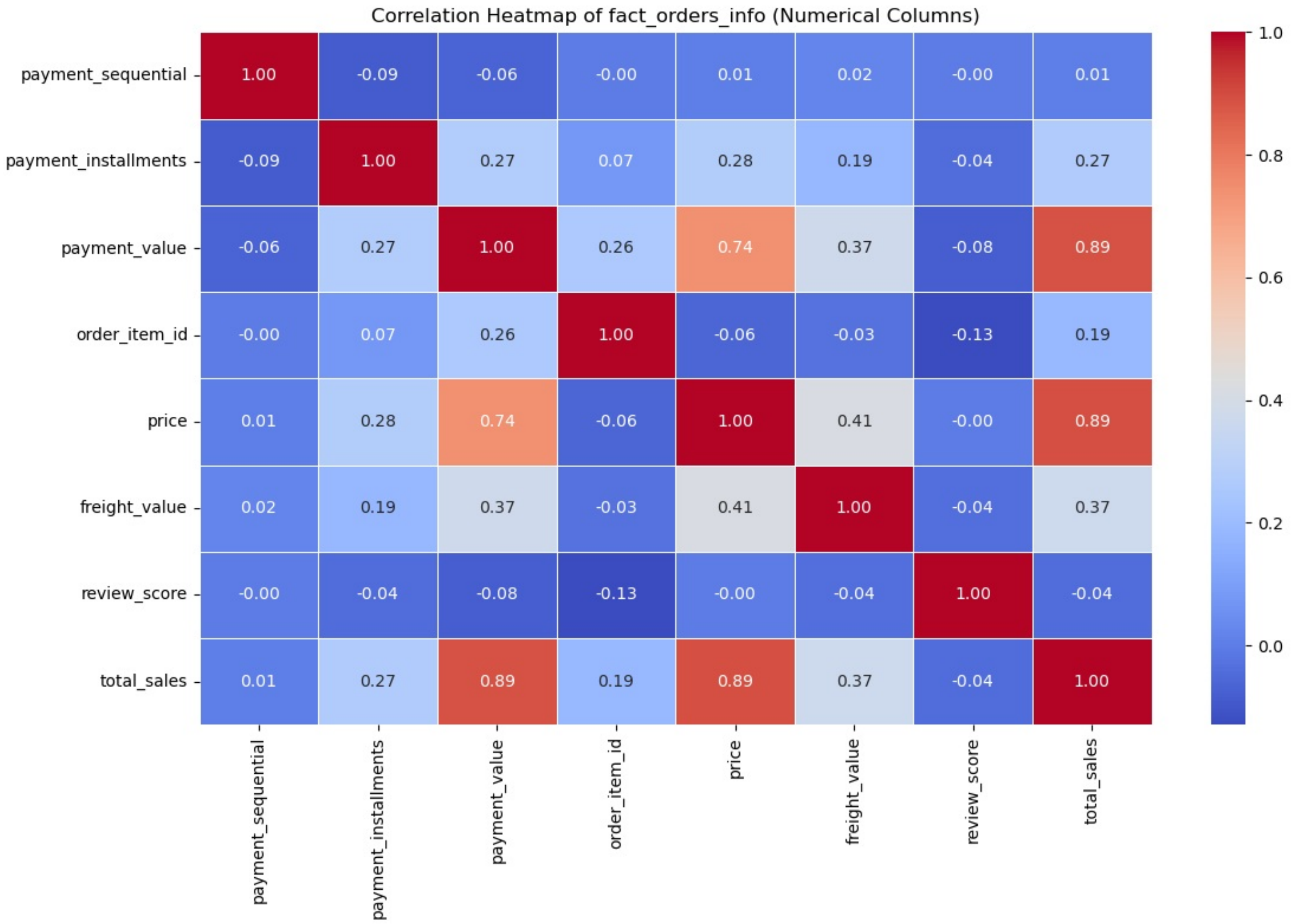




In [217…

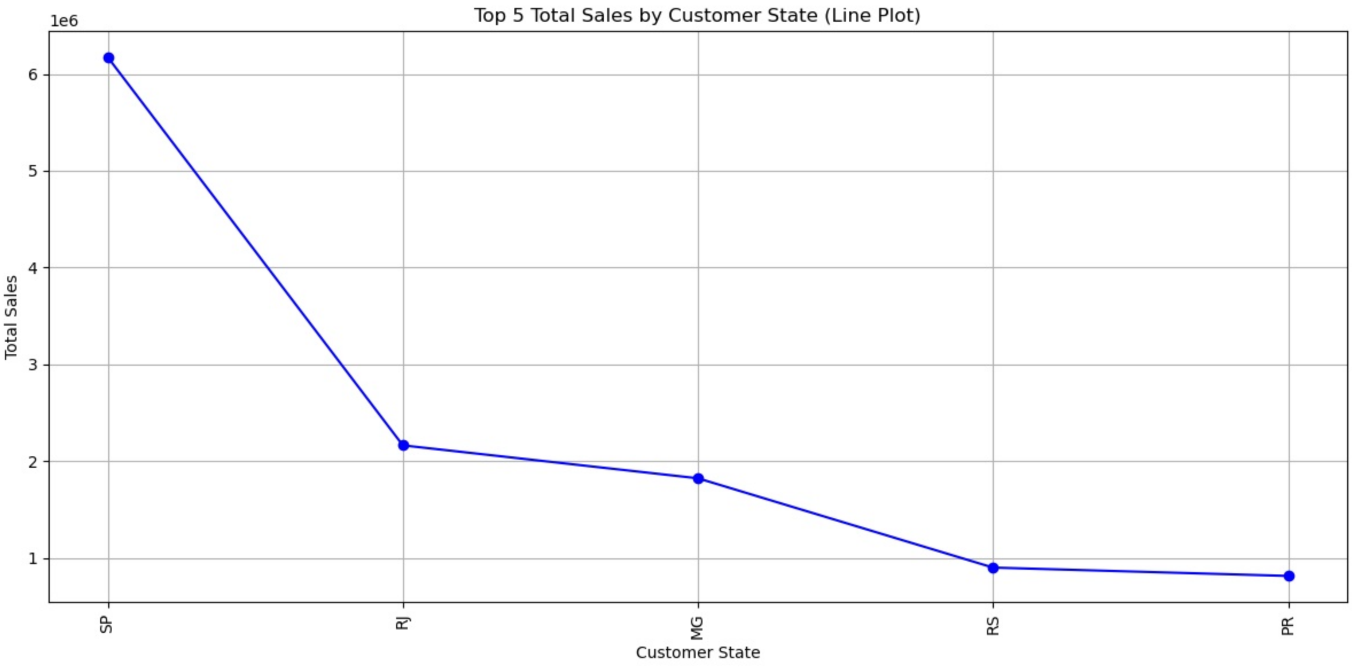
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Select numerical columns from fact_orders_info
fact_orders_numerical = fact_orders_info.select_dtypes(include=['float64', 'int64'])

# Calculate the correlation matrix
correlation_matrix = fact_orders_numerical.corr()

# Plot the correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of fact_orders_info (Numerical Columns)')
plt.tight_layout()

# Define path to save plot to the desktop
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "fact_orders_correlation_heatmap.png")
plt.savefig(desktop_path)  # Save the heatmap to the desktop
```

```python
# Show the plot
plt.show()
```



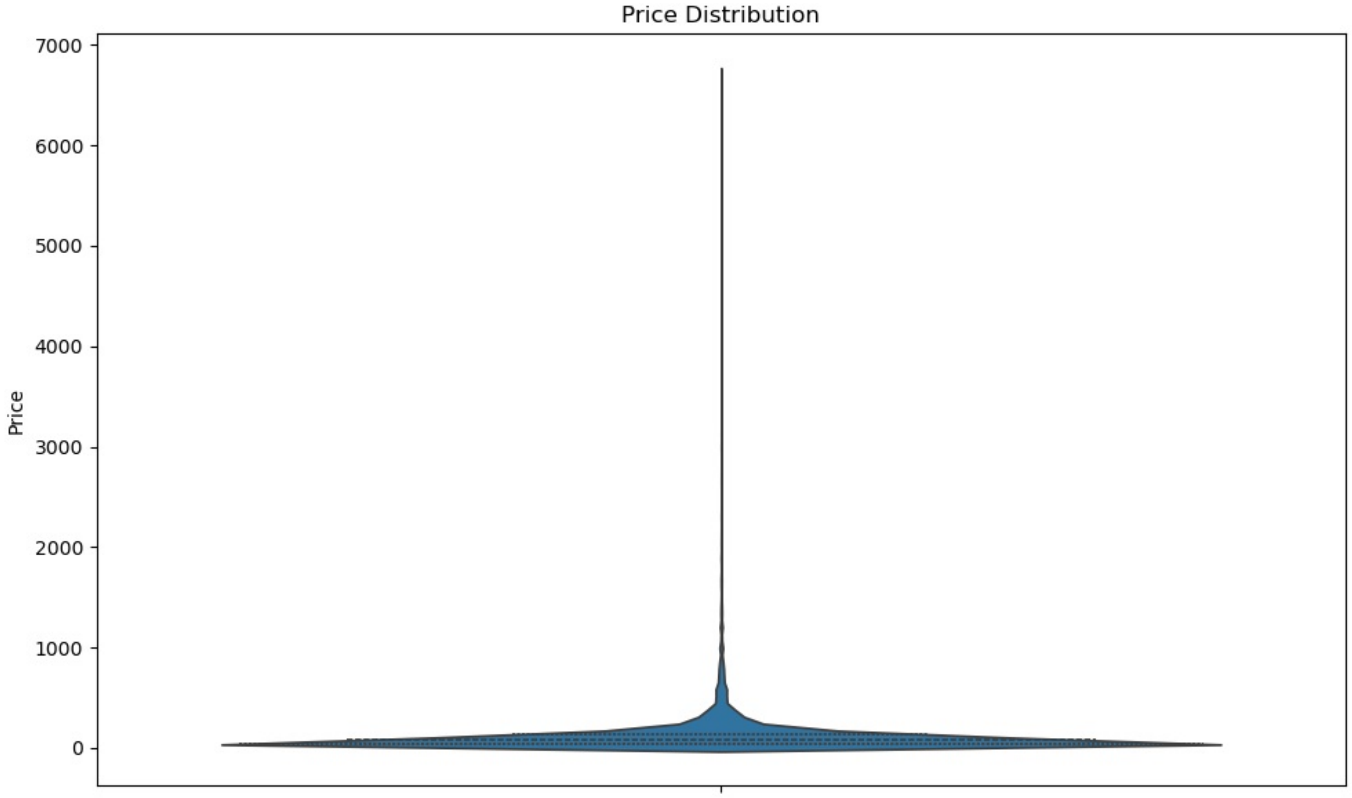
In [218…

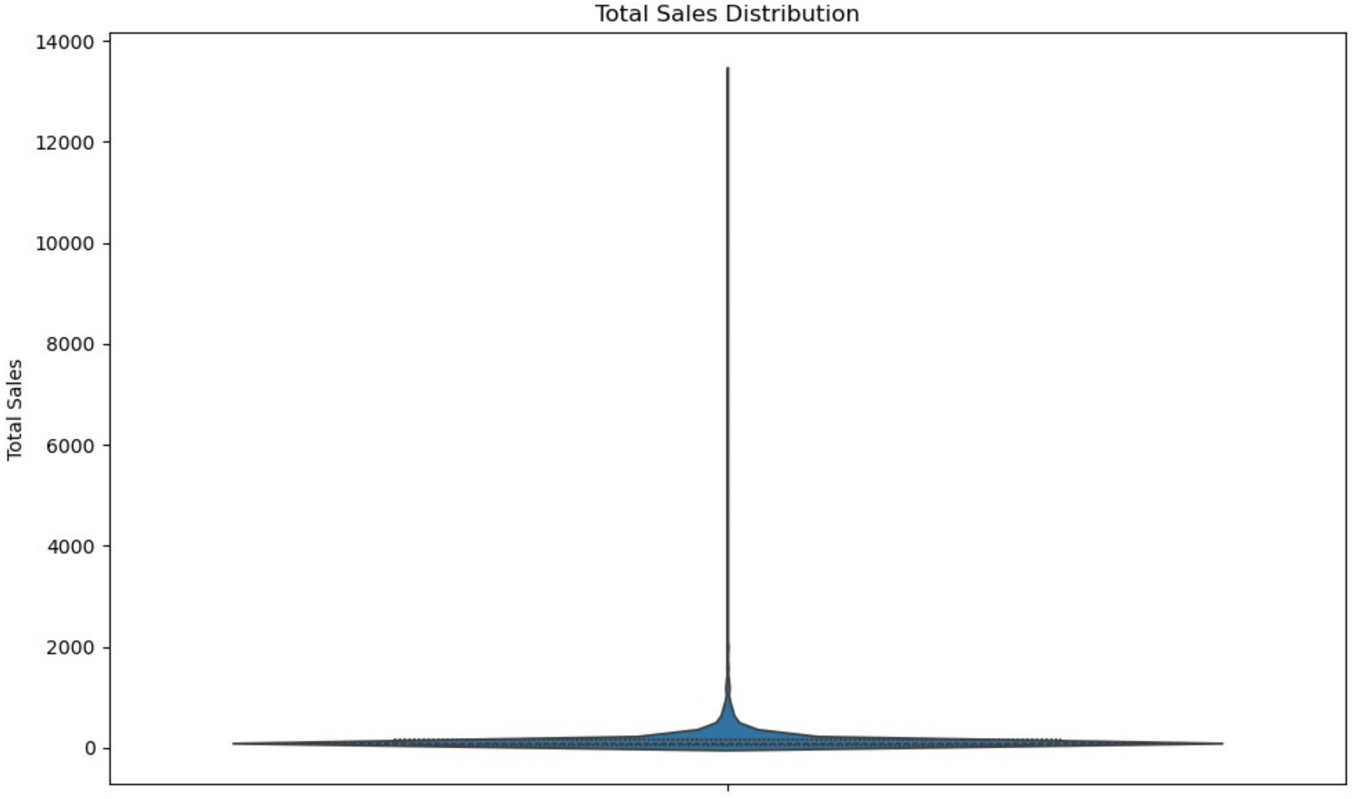
```python
import pandas as pd
import matplotlib.pyplot as plt
import os

# Step 1: Calculate total sales by order
fact_orders_info['total_sales'] = fact_orders_info['price'] * fact_orders_info['order_item_id']

# Step 2: Merge with customer state data
merged_data = pd.merge(fact_orders_info,
                       dim_customers[['customer_id', 'customer_state']],
                       on='customer_id',
                       how='left',
                       suffixes=('', '_dup'))

# Step 3: Drop duplicate columns if any
for col in merged_data.columns:
    if '_dup' in col:
        merged_data.drop(col, axis=1, inplace=True)

# Step 4: Calculate total sales by state
total_sales_by_state = merged_data.groupby('customer_state')['total_sales'].sum().reset_index()

# Step 5: Get top 5 states by total sales
top_5_states = total_sales_by_state.sort_values(by='total_sales', ascending=False).head(5)

# Step 6: Plot the line chart
plt.figure(figsize=(12, 6))
plt.plot(top_5_states['customer_state'], top_5_states['total_sales'], marker='o', color='blue')
plt.xticks(rotation=90)
plt.title('Top 5 Total Sales by Customer State (Line Plot)')
plt.ylabel('Total Sales')
plt.xlabel('Customer State')
plt.grid(True)
plt.tight_layout()

# Define path to save plot to the desktop
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop", "total_sales_by_state_line_plot.png")
plt.savefig(desktop_path)  # Save the line plot to the desktop

# Show the plot
plt.show()
```



In [219…

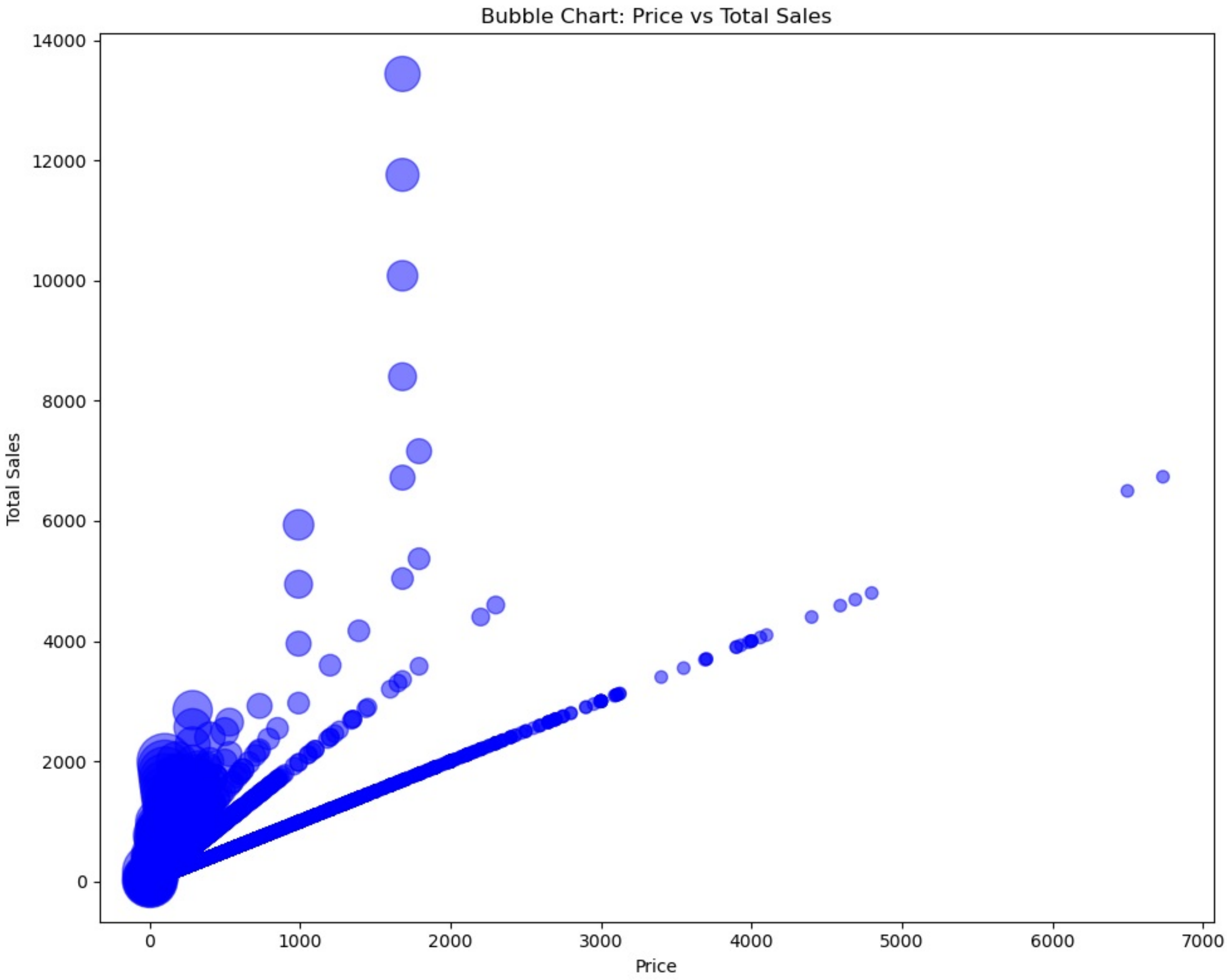
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

def plot_violin_chart(data, column, title, x_label=None, y_label=None, save_path=None):
    plt.figure(figsize=(10, 6))
    sns.violinplot(data=data, y=column, inner="quartile")
    plt.title(title)
    plt.xlabel(x_label if x_label else '')
    plt.ylabel(y_label if y_label else column)
    plt.tight_layout()

    if save_path:
        plt.savefig(save_path)
    plt.show()

# Define the desktop path
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")

# Violin plot for product price
if 'price' in fact_orders_info.columns:
    plot_violin_chart(
        fact_orders_info, 'price', 'Price Distribution', y_label='Price',
        save_path=os.path.join(desktop_path, "price_distribution.png")
    )

# Violin plot for total sales
if 'total_sales' in fact_orders_info.columns:
    plot_violin_chart(
        fact_orders_info, 'total_sales', 'Total Sales Distribution', y_label='Total Sales',
        save_path=os.path.join(desktop_path, "total_sales_distribution.png")
    )
```





In [220…

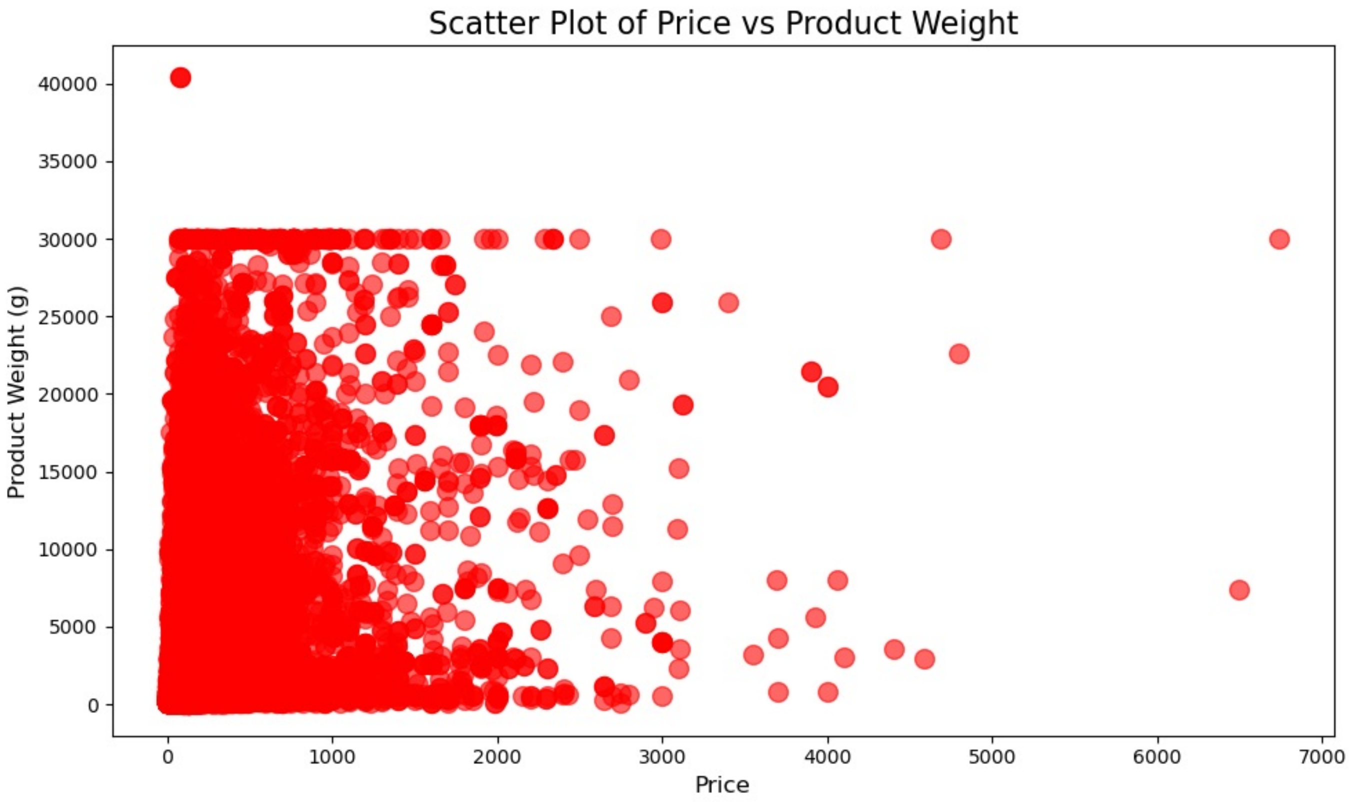
```
import matplotlib.pyplot as plt
import pandas as pd
import os

def plot_bubble_chart(data, x_column, y_column, size_column, title, x_label=None, y_label=None, size_factor=100
    plt.figure(figsize=(10, 8))

    # Scatter plot for the bubble chart
    plt.scatter(data[x_column], data[y_column],
                s=data[size_column] * size_factor,  # Bubble size is proportional to 'size_column'
                alpha=0.5, color='blue')

    # Title and axis labels
    plt.title(title)
    plt.xlabel(x_label if x_label else x_column)
    plt.ylabel(y_label if y_label else y_column)

    # Ensure the layout is neat
    plt.tight_layout()
    # Save the plot if a save path is provided
    if save_path:
        plt.savefig(save_path)
    plt.show()

# Define the desktop path
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")

# Check if columns exist in the DataFrame
if 'price' in fact_orders_info.columns and 'total_sales' in fact_orders_info.columns:
    plot_bubble_chart(fact_orders_info,
                      x_column='price',
                      y_column='total_sales',
                      size_column='order_item_id',
                      title='Bubble Chart: Price vs Total Sales',
                      x_label='Price',
                      y_label='Total Sales',
                      size_factor=50,
                      save_path=os.path.join(desktop_path, "price_vs_total_sales_bubble_chart.png"))
```



In [ ]:

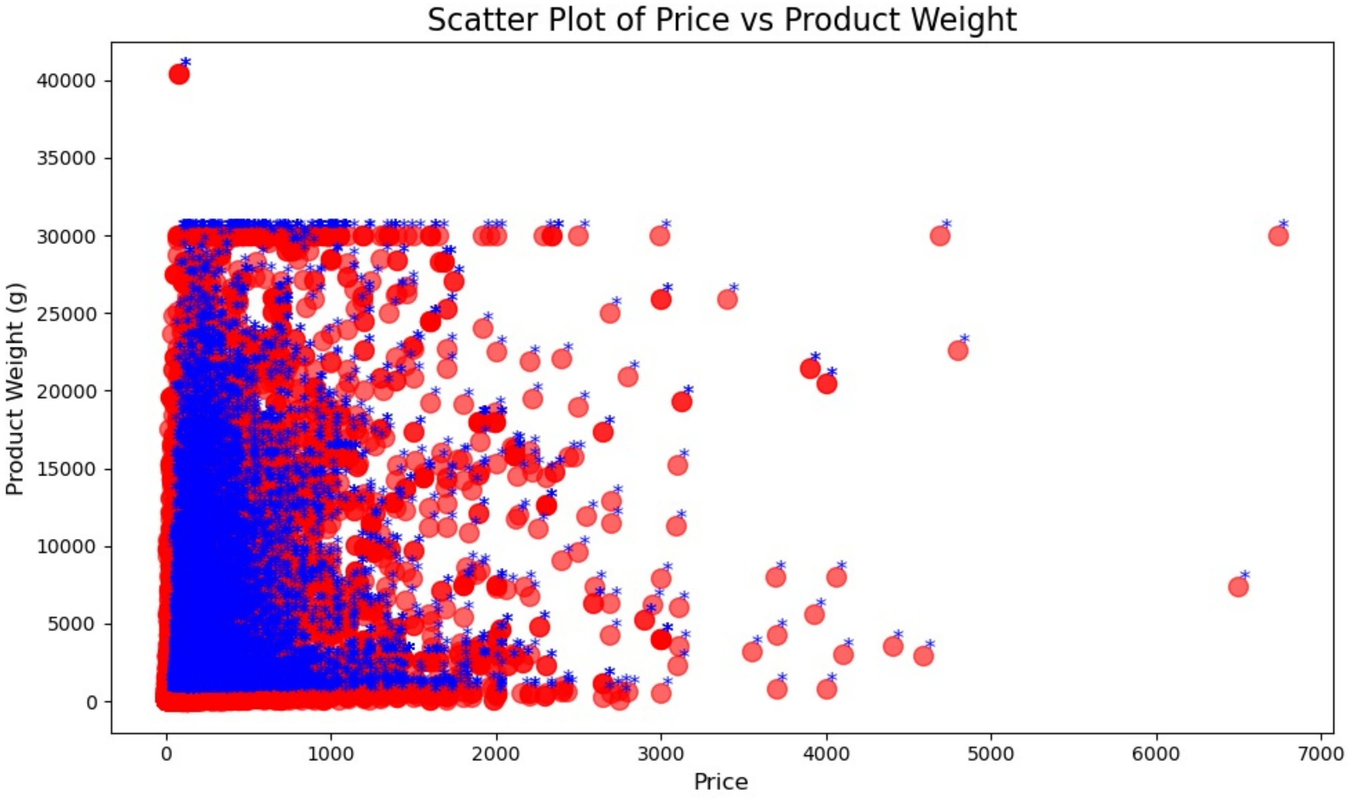
In [221…

```
import pandas as pd
import matplotlib.pyplot as plt
import os

# Merge data
merged_data = pd.merge(fact_orders_info, dim_products, on='product_id')

# Plot settings
plt.figure(figsize=(10, 6))
plt.scatter(merged_data['price'], merged_data['product_weight_g'],
            color='red', alpha=0.6, s=100)
plt.title('Scatter Plot of Price vs Product Weight', fontsize=16)
plt.xlabel('Price', fontsize=12)
plt.ylabel('Product Weight (g)', fontsize=12)
plt.tight_layout()

# Define the desktop path and save the plot
desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")
plt.savefig(os.path.join(desktop_path, "price_vs_product_weight_scatter.png"))
plt.show()
```



In [223…

```
import pandas as pd
import matplotlib.pyplot as plt
import os
merged_data = pd.merge(fact_orders_info, dim_products, on='product_id')

plt.figure(figsize=(10, 6))
plt.scatter(merged_data['price'], merged_data['product_weight_g'],
            color='red', alpha=0.6, s=100)
plt.title('Scatter Plot of Price vs Product Weight', fontsize=16)
plt.xlabel('Price', fontsize=12)
plt.ylabel('Product Weight (g)', fontsize=12)

for i, txt in enumerate(merged_data['product_id']):  # Assuming product_id for notations
    plt.annotate('*', (merged_data['price'].iloc[i], merged_data['product_weight_g'].iloc[i]),
                 fontsize=12, color='blue')
plt.tight_layout()
desktop_path = os.path.expanduser("~"), "Desktop")
plt.savefig(os.path.join(desktop_path, "price_vs_product_weight_scatter_annotated.png"))
plt.show()
```



```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js