Analyzed a Dataset of 100,000 orders spanning 2016-2018 using SQL and Python.

In [1]:
```
import pandas as pd
import numpy as np
import mysql.connector
import os

# List of CSV files and their corresponding table names.

csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv', 'order_items')
]


# Connect to the MySQL database
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='PIyush@12345',
    database='ecommerce'
)
cursor = conn.cursor()


# Folder containing the CSV files
folder_path ='D:\sql project\Ecommerce'

def get_sql_type(dtype):
    if pd.api.types.is_integer_dtype(dtype):
        return 'INT'
    elif pd.api.types.is_float_dtype(dtype):
        return 'FLOAT'
    elif pd.api.types.is_bool_dtype(dtype):
        return 'BOOLEAN'
    elif pd.api.types.is_datetime64_any_dtype(dtype):
        return 'DATETIME'
    else:
        return 'TEXT'

for csv_file, table_name in csv_files:
    file_path = os.path.join(folder_path, csv_file)

    # Read the CSV file into a pandas DataFrame
    df = pd.read_csv(file_path)

    # Replace NaN with None to handle SQL NULL
    df = df.where(pd.notnull(df), None)

    # Debugging: Check for NaN values
    print(f"Processing {csv_file}")
    print(f"NaN values before replacement:\n{df.isnull().sum()}\n")
```

```
    # Clean column names
    df.columns = [col.replace(' ', '_').replace('-', '_').replace('.', '_') for col

    # Generate the CREATE TABLE statement with appropriate data types
    columns = ', '.join([f'`{col}` {get_sql_type(df[col].dtype)}' for col in df.col
    create_table_query = f'CREATE TABLE IF NOT EXISTS `{table_name}` ({columns})'
    cursor.execute(create_table_query)

    # Insert DataFrame data into the MySQL table
    for _, row in df.iterrows():
        # Convert row to tuple and handle NaN/None explicitly
        values = tuple(None if pd.isna(x) else x for x in row)
        sql = f"INSERT INTO `{table_name}` ({', '.join(['`' + col + '`' for col in
        cursor.execute(sql, values)

    # Commit the transaction for the current CSV file
    conn.commit()

# Close the connection
conn.close()
```

```
Processing customers.csv
NaN values before replacement:
customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64

Processing orders.csv
NaN values before replacement:
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                 160
order_delivered_carrier_date     1783
order_delivered_customer_date    2965
order_estimated_delivery_date       0
dtype: int64

Processing sellers.csv
NaN values before replacement:
seller_id                 0
seller_zip_code_prefix    0
seller_city               0
seller_state              0
dtype: int64

Processing products.csv
NaN values before replacement:
product_id                      0
product category              610
product_name_length           610
product_description_length    610
product_photos_qty            610
product_weight_g                2
product_length_cm               2
product_height_cm               2
product_width_cm                2
dtype: int64

Processing geolocation.csv
NaN values before replacement:
geolocation_zip_code_prefix    0
geolocation_lat                0
geolocation_lng                0
geolocation_city               0
geolocation_state              0
dtype: int64

Processing payments.csv
NaN values before replacement:
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64

Processing order_items.csv
NaN values before replacement:
order_id               0
order_item_id          0
```

```
product_id             0
seller_id              0
shipping_limit_date    0
price                  0
freight_value          0
dtype: int64
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector

conn=mysql.connector.connect(host="localhost",
                             username="root",
                             password="PIyush@12345",
                             database="ecommerce")

cur=conn.cursor()
```

1. List all unique cities where customers are located.

In [3]:
```
query="select distinct customer_city from customers"

cur.execute(query)
output=cur.fetchall()
output
```

Out[3]:
```
[('franca',),
 ('sao bernardo do campo',),
 ('sao paulo',),
 ('mogi das cruzes',),
 ('campinas',),
 ('jaragua do sul',),
 ('timoteo',),
 ('curitiba',),
 ('belo horizonte',),
 ('montes claros',),
 ('rio de janeiro',),
 ('lencois paulista',),
 ('caxias do sul',),
 ('piracicaba',),
 ('guarulhos',),
 ('pacaja',),
 ('florianopolis',),
 ('aparecida de goiania',),
 ('santo andre',),
 ('goiania',),
 ('cachoeiro de itapemirim',),
 ('sao jose dos campos',),
 ('sao roque',),
 ('camacari',),
 ('resende',),
 ('sumare',),
 ('novo hamburgo',),
 ('sao luis',),
 ('sao jose',),
 ('santa barbara',),
 ('ribeirao preto',),
 ('ituiutaba',),
 ('taquarituba',),
 ('sao jose dos pinhais',),
 ('barrinha',),
 ('parati',),
 ('dourados',),
 ('trindade',),
 ('cascavel',),
 ('fortaleza',),
 ('brasilia',),
 ('pelotas',),
 ('porto alegre',),
 ('salto',),
 ('jundiai',),
 ('cacapava',),
 ('sao vicente',),
 ('uberlandia',),
 ('botelhos',),
 ('sao goncalo',),
 ('araucaria',),
 ('nova iguacu',),
 ('areia branca',),
 ('campos dos goytacazes',),
 ('sao carlos',),
 ('itajuba',),
 ('cruz das almas',),
 ('vassouras',),
 ('feira de santana',),
 ('niteroi',),
 ('sobral',),
 ('divinopolis',),
 ('paraiba do sul',),
 ('paulista',),
```

```
('carapicuiba',),
('bom principio',),
('astolfo dutra',),
('marialva',),
('sao jose do rio preto',),
('cabo frio',),
('contagem',),
('cafeara',),
('sao joaquim da barra',),
('foz do iguacu',),
('suzano',),
('timbo',),
('camboriu',),
('nova bassano',),
('rio grande',),
('braganca paulista',),
('barra do garcas',),
('embu',),
('urussanga',),
('silvianopolis',),
('gameleiras',),
('belem',),
('pocos de caldas',),
('santos',),
('piracaia',),
('sinop',),
('guaruja',),
('barueri',),
('feliz',),
('jambeiro',),
('ipatinga',),
('tupa',),
('blumenau',),
('moncoes',),
('balneario camboriu',),
('vargem grande',),
('rio brilhante',),
('eugenopolis',),
('paulinia',),
('apucarana',),
('recife',),
('osasco',),
('valinhos',),
('manaus',),
('cidreira',),
('santiago',),
('sao sepe',),
('alta floresta',),
('jaboatao dos guararapes',),
('ibatiba',),
('hortolandia',),
('cotia',),
('macae',),
('saudades',),
('macapa',),
('salto de pirapora',),
('taboao da serra',),
('bom jesus',),
('santa cruz do rio pardo',),
('diadema',),
('santa ines',),
('serrinha',),
('votorantim',),
('itatiaia',),
```

```
('duque de caxias',),
('varre-sai',),
('vila velha',),
('mangaratiba',),
('atibaia',),
('salvador',),
('maceio',),
('birigui',),
('petropolis',),
('sao pedro',),
('jaguariaiva',),
('franco da rocha',),
('pindamonhangaba',),
('irati',),
('ribeira',),
('barbacena',),
('limoeiro',),
('espigao do oeste',),
('belford roxo',),
('coronel fabriciano',),
('itanhaem',),
('bebedouro',),
('americana',),
('uba',),
('arapongas',),
('vinhedo',),
('itajai',),
('bauru',),
('pradopolis',),
('aripuana',),
('guaratinga',),
('ponta pora',),
('aracatuba',),
('volta redonda',),
('maringa',),
('araraquara',),
('matipo',),
('santo antonio da patrulha',),
('barra mansa',),
('diamantina',),
('mairinque',),
('capitao leonidas marques',),
('sao sebastiao do paraiso',),
('rosario do sul',),
('itaguai',),
('paraopeba',),
('guarapuava',),
('crisolita',),
('pirai',),
('linhares',),
('agudos',),
('sao joao de meriti',),
('navegantes',),
('pirassununga',),
('faxinal dos guedes',),
('criciuma',),
('nova venecia',),
('passo fundo',),
('ibia',),
('manhuacu',),
('altamira',),
('aperibe',),
('cuiaba',),
('canoas',),
```

```
('cubatao',),
('campo limpo paulista',),
('cambe',),
('itaquaquecetuba',),
('sao caetano do sul',),
('sao goncalo do rio abaixo',),
('rolandia',),
('tres coracoes',),
('cacapava do sul',),
('sao joao nepomuceno',),
('leme',),
('araras',),
('cortes',),
('brusque',),
('montenegro',),
('itaberai',),
('santa rosa de viterbo',),
('agua fria de goias',),
('marau',),
('curvelo',),
('juiz de fora',),
('mogi-guacu',),
('guaratingueta',),
('paranagua',),
('lins',),
('campo bom',),
('sertaozinho',),
('tres lagoas',),
('jau',),
('campos de julio',),
('saquarema',),
('artur nogueira',),
('uaua',),
('jandira',),
('concordia',),
('nova friburgo',),
('sorocaba',),
('ponte nova',),
('araquari',),
('muriae',),
('nova lima',),
('inhumas',),
('italva',),
('tres rios',),
('santa maria',),
('itagiba',),
('paracatu',),
('xaxim',),
('laranjeiras do sul',),
('itapiuna',),
('formosa',),
('ivoti',),
('juazeiro',),
('ponta grossa',),
('campina grande',),
('maua',),
('salgueiro',),
('lorena',),
('toledo',),
('sao pedro da aldeia',),
('vianopolis',),
('arapiraca',),
('porto seguro',),
('ariquemes',),
```

```
('presidente getulio',),
('rio negro',),
('ribeirao pires',),
('sao jose da coroa grande',),
('agua doce do norte',),
('anapolis',),
('guararapes',),
('farroupilha',),
('almenara',),
('rio das ostras',),
('gravatai',),
('brumado',),
('marilia',),
('itabira',),
('claudia',),
('para de minas',),
('miguelopolis',),
('terra roxa',),
('araguari',),
('lages',),
('embu das artes',),
('limeira',),
('taubate',),
('santa fe do sul',),
('caieiras',),
('carangola',),
('chapada do norte',),
('loanda',),
('passa tres',),
('aracoiaba da serra',),
('itaborai',),
('vitoria',),
('sao bento do sul',),
('indaiatuba',),
('boituva',),
('teresopolis',),
('pinhalzinho',),
('petrolina',),
('natal',),
('barreiras',),
('januaria',),
('ipiabas',),
('firminopolis',),
('joinville',),
('mococa',),
('valenca',),
('sao miguel do oeste',),
('jales',),
('rio formoso',),
('angra dos reis',),
('alfredo chaves',),
('itapetinga',),
('gurupi',),
('nucleo residencial pilar',),
('coromandel',),
('charqueada',),
('itau de minas',),
('ibiruba',),
('bertioga',),
('ipiau',),
('matozinhos',),
('teresina',),
('entre rios',),
('juina',),
```

```
('mairipora',),
('marechal candido rondon',),
('avare',),
('icem',),
('sao sebastiao',),
('serra',),
('mirassol',),
('taperuaba',),
('alfenas',),
('leopoldina',),
('sao joao da boa vista',),
('parnamirim',),
('teixeira de freitas',),
('joao pessoa',),
('chacara',),
('rio novo do sul',),
('guapore',),
('rio branco',),
('ferraz de vasconcelos',),
('petrolandia',),
('araruna',),
('gravata',),
('alegre',),
('londrina',),
('senhor do bonfim',),
('santana de parnaiba',),
('piratininga',),
('monte carmelo',),
('ipiranga',),
('jacarei',),
('picos',),
('una',),
('taguai',),
('peabiru',),
('conquista',),
('bicas',),
('forquilha',),
('guaicara',),
('santa vitoria',),
('adamantina',),
('rio claro',),
('sao jose de uba',),
('sorriso',),
('aracaju',),
('conselheiro lafaiete',),
('taguatinga',),
('tiangua',),
('paranaiba',),
('david canabarro',),
('itarare',),
('cajueiro',),
('cruzeiro',),
('santa cruz do sul',),
('presidente epitacio',),
('castanhal',),
('cariacica',),
('bom jesus dos perdoes',),
('urucurituba',),
('olinda',),
('sao jose do rio pardo',),
('guapimirim',),
('vargem grande paulista',),
('chapeco',),
('videira',),
```

```
('canapi',),
('pontal do parana',),
('bento goncalves',),
('caceres',),
('bituruna',),
('sao bento do sapucai',),
('mage',),
('forquilhinha',),
('itaberaba',),
('gramado',),
('guacui',),
('pato branco',),
('itanhem',),
('palmas',),
('bage',),
('francisco morato',),
('cosmopolis',),
('carmo',),
('armacao dos buzios',),
('santa maria da vitoria',),
('pedro leopoldo',),
('tres marias',),
('santo antonio de padua',),
('porto feliz',),
('gaspar',),
('palhoca',),
('pouso alegre',),
('guaiuba',),
('fraiburgo',),
('aurea',),
('botucatu',),
('corumba',),
('cajamar',),
('queimados',),
('sao jorge do ivai',),
('nova laranjeiras',),
('guaxupe',),
('osvaldo cruz',),
('ivora',),
('anaurilandia',),
('caraguatatuba',),
('regeneracao',),
('bilac',),
('faxinal',),
('pedro velho',),
('uberaba',),
('viamao',),
('cachoeira de minas',),
('coelho neto',),
('itajobi',),
('carmo do rio claro',),
('marica',),
('campo novo do parecis',),
('boa vista',),
('ipameri',),
('carlos barbosa',),
('anicuns',),
('rio bananal',),
('lavras da mangabeira',),
('balsamo',),
('guariba',),
('sao vendelino',),
('campo mourao',),
('aracati',),
```

```
('santo antonio do descoberto',),
('santo amaro da imperatriz',),
('betim',),
('vicosa',),
('dom eliseu',),
('campo largo',),
('santa rosa de lima',),
('quissama',),
('mirandopolis',),
('itapevi',),
('francisco beltrao',),
("dias d'avila",),
('novo horizonte',),
('poa',),
('querencia',),
('campo grande',),
('pedreira',),
('pariquera-acu',),
('itabaianinha',),
('pitangui',),
('araruama',),
('campo formoso',),
('guanambi',),
('itapolis',),
('pederneiras',),
('itabirito',),
('itaipava',),
('arroio do sal',),
('glaura',),
('lagoa vermelha',),
('mantena',),
('peruibe',),
('tubarao',),
('bonfim',),
('alvorada',),
('patrocinio',),
('mineiros',),
('itabuna',),
('tijucas',),
('santo antonio de jesus',),
('canarana',),
('itapecerica da serra',),
('nova prata do iguacu',),
('treze tilias',),
('anchieta',),
('quatigua',),
('iturama',),
('tres de maio',),
('santa rita do passa quatro',),
('porto esperidiao',),
('bonfim paulista',),
('cajuru',),
('cacador',),
('paragominas',),
('boa esperanca do sul',),
('tres pontas',),
('juazeiro do norte',),
('bambui',),
('comendador levy gasparian',),
('sao miguel do aleixo',),
('palmeira dos indios',),
('cerquilho',),
('saudade do iguacu',),
('ipaussu',),
```

```
('alto paraiso de goias',),
('santa isabel',),
('quirinopolis',),
('caruaru',),
('casimiro de abreu',),
('chavantes',),
('ecoporanga',),
('cachoeira paulista',),
('rio verde',),
('maracanau',),
('canoinhas',),
('ituverava',),
('buriti dos lopes',),
('rio do antonio',),
('guaranta',),
('santa luzia',),
('tucurui',),
('paranavai',),
('pinhais',),
('governador valadares',),
('trajano de moraes',),
('caete',),
('abaetetuba',),
('lavras',),
('coronel joao sa',),
('divino',),
('macaubas',),
('lucelia',),
('brejo da madre de deus',),
('carai',),
('corbelia',),
('varginha',),
('ouro preto',),
('prado',),
('colniza',),
('piuma',),
('rancharia',),
('barretos',),
('eunapolis',),
('floresta',),
('sao jose do norte',),
('rio bonito',),
('vargem bonita',),
('monte mor',),
('valente',),
('balsas',),
('nhandeara',),
('lauro de freitas',),
('osorio',),
('machado',),
('sao leopoldo',),
('soledade',),
('assis',),
('guarapari',),
('santo antonio do caiua',),
('esteio',),
('cambuci',),
('campina grande do sul',),
('baixo guandu',),
('fernao',),
('ibitinga',),
('santa cruz das palmeiras',),
('vila muriqui',),
('sao mateus',),
```

```
('piracuruca',),
('cordeiro',),
('cachoeira do sul',),
('barra do pirai',),
('castro',),
('ico',),
('sete lagoas',),
('itumbiara',),
('aluminio',),
('jatai',),
('frederico westphalen',),
('goioere',),
('sao jose da tapera',),
('porto uniao',),
('cocalinho',),
('curitibanos',),
('campos novos',),
('ubatuba',),
('monte santo de minas',),
('ribeirao das neves',),
('sao simao',),
('urutai',),
('itaobim',),
('liberdade',),
('arcos',),
('ribeirao',),
('patos de minas',),
('penapolis',),
('eusebio',),
('cordeiros',),
('victor graeff',),
('santa rita do araguaia',),
('poxoreu',),
('ananas',),
('conceicao dos ouros',),
('alagoinhas',),
('sao joao da barra',),
('lindoia',),
('bonfinopolis',),
('ibiam',),
('morungaba',),
('andradina',),
('tatui',),
('mata verde',),
('cornelio procopio',),
('itapipoca',),
('queluz',),
('godoy moreira',),
('iracemapolis',),
('buritizeiro',),
('xique-xique',),
('uruacu',),
('japi',),
('itauna',),
('sao francisco de assis',),
('pitangueiras',),
('ze doca',),
('dracena',),
('cachoeiras de macacu',),
('barauna',),
('formiga',),
('rodeio',),
('jequie',),
('juvenilia',),
```

```
('poco fundo',),
('amparo',),
('lauro muller',),
('japeri',),
('ourinhos',),
('lagoa dos gatos',),
('embu-guacu',),
('nossa senhora do remedio',),
('tambau',),
('orlandia',),
('bananeiras',),
('sao joao do manhuacu',),
('sarandi',),
('congonhas',),
('boa esperanca',),
('sandolandia',),
('garca',),
('descalvado',),
('guaracai',),
('aruana',),
('inga',),
('taio',),
('gaurama',),
('praia grande',),
('janauba',),
('nova monte verde',),
('mongagua',),
('monte alto',),
('sao joao do piaui',),
('primavera do leste',),
('seropedica',),
('indaial',),
('teofilo otoni',),
('santa terezinha',),
('planaltina',),
('caravelas',),
('muritiba',),
('itatiba',),
('piumhii',),
('pitanga',),
('capelinha',),
('ipero',),
('tarabai',),
('aparecida do taboado',),
('maioba',),
('monnerat',),
('sao lourenco do sul',),
('biguacu',),
('venda nova do imigrante',),
('arapoti',),
('fazenda rio grande',),
('senges',),
('aracruz',),
('canela',),
('porto franco',),
('veranopolis',),
('candiota',),
('carmo do paranaiba',),
('itacare',),
('desembargador otoni',),
('joanopolis',),
('registro',),
('ipiranga do norte',),
('felipe guerra',),
```

```
('dumont',),
('itapetininga',),
('bom despacho',),
('lagoa santa',),
('estacao',),
('panorama',),
('palmares',),
('prata',),
('santa adelia',),
('iguaba grande',),
('sarzedo',),
('dourado',),
('alcinopolis',),
('limoeiro do norte',),
('ibiraci',),
('ilhabela',),
('icara',),
('paiva',),
('batatais',),
('formoso do araguaia',),
('itapaci',),
('fernandopolis',),
('miguel pereira',),
('tocos',),
('campos do jordao',),
('catalao',),
('sao jose da lapa',),
('marituba',),
('arapora',),
('turvo',),
('rubiataba',),
('guaruja do sul',),
('pedra bela',),
('perdizes',),
('jesuania',),
('antonio carlos',),
('itapecerica',),
('monte belo',),
('mossoro',),
('piraju',),
('encruzilhada do sul',),
('igaratinga',),
('sao luis de montes belos',),
('santa rosa',),
('lagoinha',),
('redentora',),
('cuite',),
('coxim',),
('lucas do rio verde',),
('ibirataia',),
('olimpia',),
('cedro',),
('mario campos',),
('umuarama',),
('nova xavantina',),
('sao borja',),
('aguai',),
('itapira',),
('uniao da vitoria',),
('nova odessa',),
('itapuranga',),
('nova cruz',),
('apuiares',),
('almirante tamandare',),
```

```
('correia pinto',),
('itamarandiba',),
('juscimeira',),
('ipora',),
('cedro de sao joao',),
("santa barbara d'oeste",),
('sao joao da urtiga',),
('valparaiso de goias',),
('jussara',),
('capao da canoa',),
('araxa',),
('campo belo',),
('monte aprazivel',),
('bom jesus do querendo',),
('socorro',),
('catanduva',),
('monte castelo',),
('tabatinga',),
('bayeux',),
('ijui',),
('tangara da serra',),
('campos borges',),
('palma',),
('minacu',),
('miracema',),
('taruma',),
('sao francisco do sul',),
('cedral',),
('camaragibe',),
('lajeado',),
('cerqueira cesar',),
('frutal',),
('igarata',),
('novo gama',),
('sapucaia do sul',),
('juquitiba',),
('terra boa',),
('joao monlevade',),
('porangatu',),
('formosa da serra negra',),
('rafard',),
('rainha do mar',),
('pontal',),
('rio do sul',),
('nilopolis',),
('maracaju',),
('santo augusto',),
('banabuiu',),
('buenopolis',),
('barbalha',),
('quintana',),
('chorrocho',),
('votuporanga',),
('poco verde',),
('cipo-guacu',),
('santarem',),
('santa clara do sul',),
('ibiuna',),
('capivari',),
('arraias',),
('patos',),
('sao francisco do guapore',),
('vitoria da conquista',),
('urucuca',),
```

```
('vacaria',),
('campos altos',),
('nossa senhora do socorro',),
('cruzeiro do sul',),
('cataguases',),
('varzea grande',),
('santa rita do sapucai',),
('jaboticabal',),
('ilha comprida',),
('brasilia de minas',),
('moreno',),
('capanema',),
('tombos',),
('espumoso',),
('ouro branco',),
('valenca do piaui',),
('igrejinha',),
('engenheiro coelho',),
('parauapebas',),
('afonso claudio',),
('sao bernardo',),
('salto do jacui',),
('pires do rio',),
('astorga',),
('medina',),
('porto ferreira',),
('aracariguama',),
('hidrolandia',),
('virginopolis',),
('soledade de minas',),
('casca',),
('sapiranga',),
('angatuba',),
('ibirapua',),
('itupeva',),
('espera feliz',),
('penedo',),
('ibirite',),
('miracatu',),
('francisco santos',),
('paracambi',),
('cardoso',),
('floriano',),
('rondonopolis',),
('barra de sao francisco',),
('lavinia',),
('guajara-mirim',),
('guimaraes',),
('sao tome',),
('medianeira',),
('conceicao das pedras',),
('rinopolis',),
('capim grosso',),
('guaira',),
('euclides da cunha paulista',),
('lagoa da prata',),
('nanuque',),
('luis antonio',),
('senador firmino',),
('viradouro',),
('itabera',),
('cambara',),
('sananduva',),
('tapera',),
```

```
('tupaciguara',),
('colombo',),
('ananindeua',),
('coribe',),
('rio doce',),
('mogi mirim',),
('tocos do moji',),
('beberibe',),
('formosa do rio preto',),
('claudio',),
('rio paranaiba',),
('humberto de campos',),
('camocim de sao felix',),
('santo antonio de posse',),
('mesquita',),
('passa quatro',),
('belo oriente',),
('alto araguaia',),
('igarassu',),
('porto velho',),
('cajazeiras',),
('garopaba',),
('rio azul',),
('dois corregos',),
('sooretama',),
('cruzeiro do oeste',),
('nazare paulista',),
('riversul',),
('iguaracu',),
('maracas',),
('parana',),
('reboucas',),
('ametista do sul',),
('itu',),
('cacu',),
('vespasiano',),
('lindolfo collor',),
('sao lourenco',),
('santana do sobrado',),
('cafelandia',),
('ibiapina',),
('schroeder',),
('sao joao do oriente',),
('matao',),
('itanhandu',),
('santo angelo',),
('capinopolis',),
('colatina',),
('nova mutum',),
('munhoz de melo',),
('ibate',),
('unai',),
('medeiros neto',),
('anta',),
('ouroeste',),
('central',),
('nova independencia',),
('itapema',),
('sao jose do cedro',),
('camutanga',),
('ipumirim',),
('cristalia',),
('andira',),
('panelas',),
```

```
('comodoro',),
('domingos martins',),
('tangua',),
('sao gotardo',),
('araguaina',),
('ilheus',),
('louveira',),
('imbituba',),
('ouricuri',),
('itatinga',),
('santo cristo',),
("arraial d'ajuda",),
('piracanjuba',),
('russas',),
('lambari',),
('vargem alta',),
('sacra familia do tingua',),
('assis chateaubriand',),
('lago da pedra',),
('unistalda',),
('sao joao evangelista',),
('paramirim',),
('goias',),
('paracuru',),
('cruz alta',),
('presidente bernardes',),
('pains',),
('sao goncalo do rio preto',),
('jacinto machado',),
('sobralia',),
('maraba',),
('paraibuna',),
('toropi',),
('jauru',),
('cristalandia',),
('castelo do piaui',),
('ibipora',),
('presidente dutra',),
('vera cruz',),
('paulo frontin',),
...]
```

2. Count the number of orders placed in 2017.

In [4]:
```
query="select count(order_id) from orders where year(order_purchase_timestamp)=2017

cur.execute(query)
output=cur.fetchall()
orders=output[0][0]
print('total orders placed in 2017 is' ,orders)
```

```
total orders placed in 2017 is 90202
```

3. Find the total sales per category.

In [5]:
```
query = """ select upper(products.product_category) category,
round(sum(payments.payment_value),2) sales
from products join order_items
```

```
on products.product_id = order_items.product_id
join payments
on payments.order_id = order_items.order_id
group by category
"""

cur.execute(query)
output=cur.fetchall()
output


df = pd.DataFrame(output, columns = ["Category", "Sales"])
df
```

Out[5]:

	Category	Sales
0	PERFUMERY	4053909.28
1	FURNITURE DECORATION	11441411.13
2	TELEPHONY	3895056.41
3	BED TABLE BATH	13700429.37
4	AUTOMOTIVE	6818354.65
...	...	...
69	CDS MUSIC DVDS	9595.44
70	LA CUISINE	23308.24
71	FASHION CHILDREN'S CLOTHING	6285.36
72	PC GAMER	17395.44
73	INSURANCE AND SERVICES	2596.08

74 rows × 2 columns

4. Calculate the percentage of orders that were paid in installments.

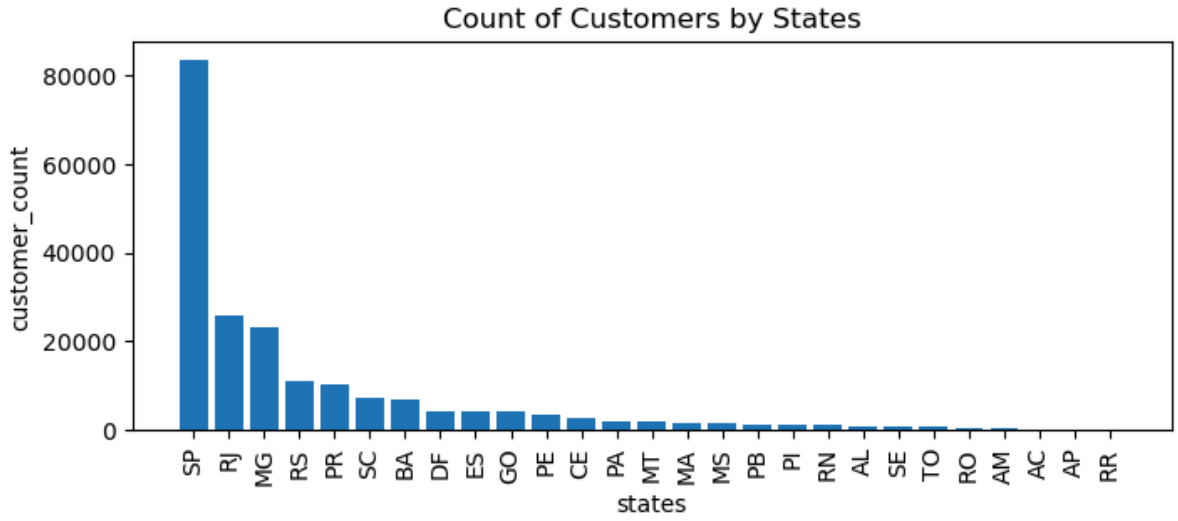
In [6]:

```
query = """ select ((sum(case when payment_installments >= 1 then 1
else 0 end))/count(*))*100 from payments
"""

cur.execute(query)

output = cur.fetchall()
data= output[0][0]

print("the percentage of orders that were paid in installments is", data)
```

```
the percentage of orders that were paid in installments is 99.9981
```

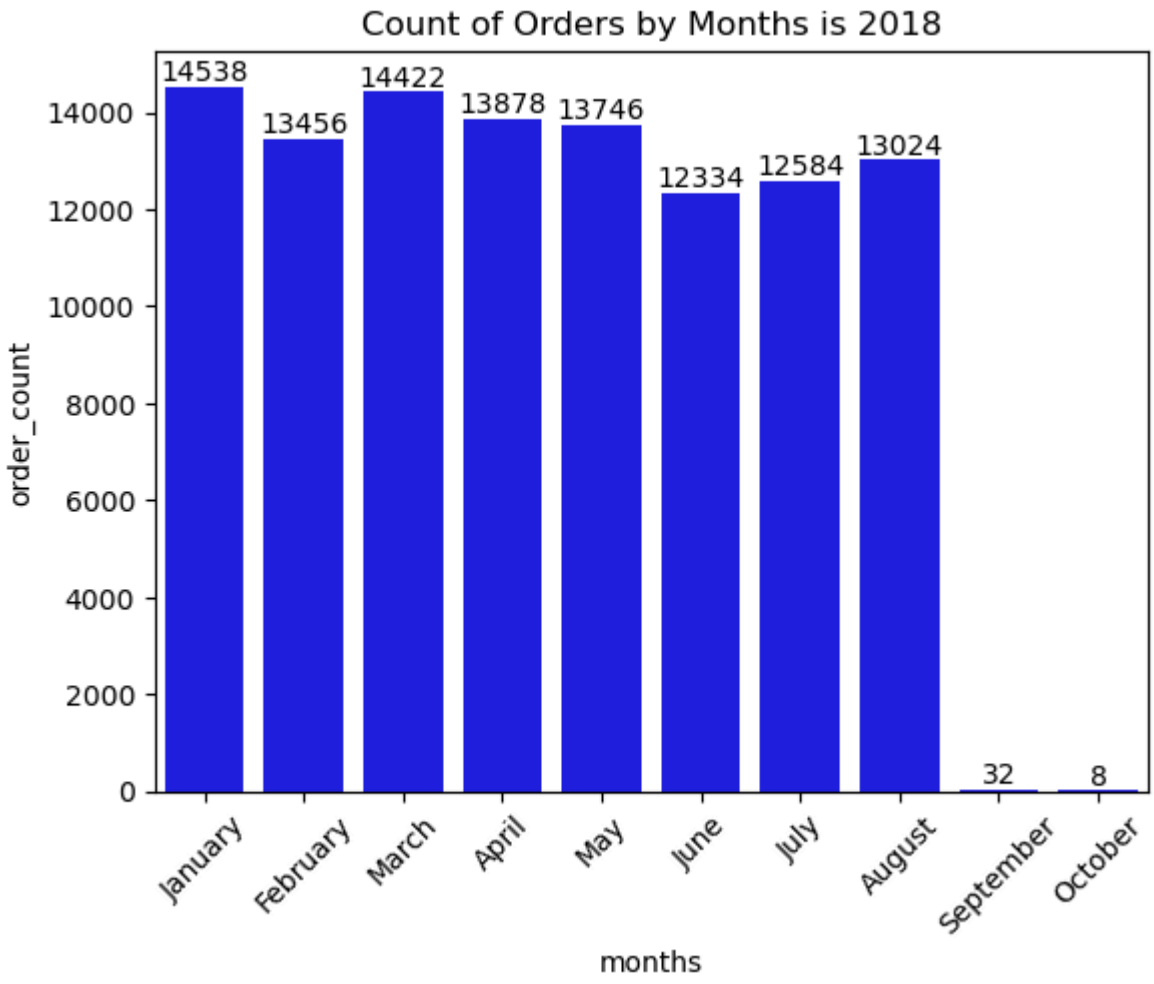
5. Count the number of customers from each state.

In [7]:
```
query = """ select customer_state ,count(customer_id)
from customers group by customer_state
"""

cur.execute(query)

output = cur.fetchall()
df = pd.DataFrame(output, columns = ["state", "customer_count" ])
df = df.sort_values(by = "customer_count", ascending= False)
plt.figure(figsize = (8,3))
plt.bar(df["state"], df["customer_count"])
plt.xticks(rotation = 90)
plt.xlabel("states")
plt.ylabel("customer_count")
plt.title("Count of Customers by States")
plt.show()
```



6. Calculate the number of orders per month in 2018.

In [8]:
```
query = """ select monthname(order_purchase_timestamp) months, count(order_id) orde
from orders where year(order_purchase_timestamp) = 2018
group by months
"""

cur.execute(query)

data = cur.fetchall()
df = pd.DataFrame(data, columns = ["months", "order_count"])
months = ["January", "February","March","April","May","June","July","August","Septe

ax = sns.barplot(x = df["months"],y =  df["order_count"], data = df, order = months
plt.xticks(rotation = 45)
ax.bar_label(ax.containers[0])
plt.title("Count of Orders by Months is 2018")

plt.show()
```



7. Find the average number of products per order, grouped by customer city.

In [9]:
```
query = """with count_per_order as
(select orders.order_id, orders.customer_id, count(order_items.order_id) as oc
from orders join order_items
on orders.order_id = order_items.order_id
group by orders.order_id, orders.customer_id)

select customers.customer_city, round(avg(count_per_order.oc),2) average_orders
from customers join count_per_order
on customers.customer_id = count_per_order.customer_id
group by customers.customer_city order by average_orders desc
"""

cur.execute(query)

output = cur.fetchall()
df = pd.DataFrame(output,columns = ["customer city", "average products/order"])
df.head(10)
```

Out[9]:

	customer city	average products/order
0	padre carvalho	28.00
1	celso ramos	26.00
2	datas	24.00
3	candido godoi	24.00
4	matias olimpio	20.00
5	cidelandia	16.00
6	curralinho	16.00
7	picarra	16.00
8	morro de sao paulo	16.00
9	teixeira soares	16.00

8. Calculate the percentage of total revenue contributed by each product category.

In [11]:
```
query = """select upper(products.product_category) category,
round((sum(payments.payment_value)/(select sum(payment_value) from payments))*100,2
from products join order_items
on products.product_id = order_items.product_id
join payments
on payments.order_id = order_items.order_id
group by category order by sales_percentage desc"""


cur.execute(query)
output = cur.fetchall()
df = pd.DataFrame(output,columns = ["Category", "percentage distribution"])
df.head()
```

Out[11]:

	Category	percentage distribution
0	BED TABLE BATH	42.79
1	HEALTH BEAUTY	41.41
2	COMPUTER ACCESSORIES	39.61
3	FURNITURE DECORATION	35.73
4	WATCHES PRESENT	35.71

9.Identify the correlation between product price and the number of times a product has been purchased.

In [24]:
```
import pandas as pd
import numpy as np
```

```
import mysql.connector
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

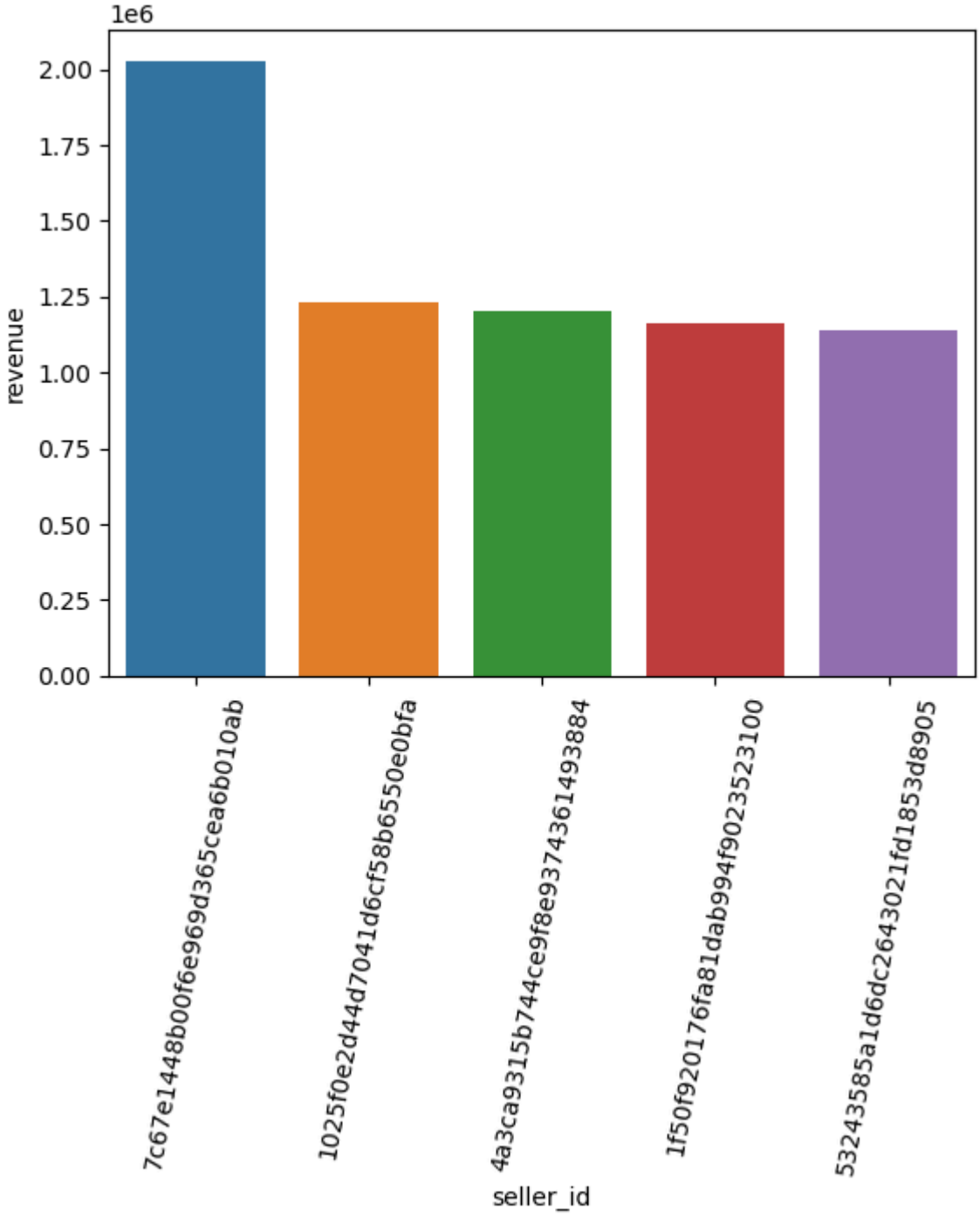
In [26]:
```
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='PIyush@12345',
    database='ecommerce'
)
cur = conn.cursor()
```

In [28]:
```
query = """select products.product_category,
count(order_items.product_id),
round(avg(order_items.price),2)
from products join order_items
on products.product_id = order_items.product_id
group by products.product_category"""

cur.execute(query)
output=cur.fetchall()
df = pd.DataFrame(output,columns = ["Category", "order_count","price"])

arr1 = df["order_count"]
arr2 = df["price"]

a = np.corrcoef([arr1,arr2])
print("the correlation is", a[0][-1])
```

```
the correlation is -0.10631514167157562
```

10.Calculate the total revenue generated by each seller, and rank them by revenue.

In [32]:
```
query = """ select *, dense_rank() over(order by revenue desc) as rn from
(select order_items.seller_id, sum(payments.payment_value)
revenue from order_items join payments
on order_items.order_id = payments.order_id
group by order_items.seller_id) as a """

cur.execute(query)
output = cur.fetchall()
df = pd.DataFrame(output, columns = ["seller_id", "revenue", "rank"])
df = df.head()
sns.barplot(x = "seller_id", y = "revenue", data = df)
plt.xticks(rotation = 80)
plt.show()
```



11.Calculate the moving average of order values for each customer over their order history.

In [34]:
```
query = """select years, months , payment, sum(payment)
over(order by years, months) cumulative_sales from
(select year(orders.order_purchase_timestamp) as years,
month(orders.order_purchase_timestamp) as months,
round(sum(payments.payment_value),2) as payment from orders join payments
on orders.order_id = payments.order_id
group by years, months order by years, months) as a
"""
cur.execute(query)
output = cur.fetchall()
```

```
df = pd.DataFrame(output)
df
```

Out[34]:

	0	1	2	3
0	2016	9	1008.96	1008.96
1	2016	10	236361.92	237370.88
2	2016	12	78.48	237449.36
3	2017	1	553952.16	791401.52
4	2017	2	1167632.04	1959033.56
5	2017	3	1799454.40	3758487.96
6	2017	4	1671152.12	5429640.08
7	2017	5	2371675.28	7801315.36
8	2017	6	2045105.52	9846420.88
9	2017	7	2369531.68	12215952.56
10	2017	8	2697585.28	14913537.84
11	2017	9	2911049.80	17824587.64
12	2017	10	3118711.52	20943299.16
13	2017	11	4779531.20	25722830.36
14	2017	12	3513605.92	29236436.28
15	2018	1	4460016.72	33696453.00
16	2018	2	3969853.36	37666306.36
17	2018	3	4638608.48	42304914.84
18	2018	4	4643141.92	46948056.76
19	2018	5	4615928.60	51563985.36
20	2018	6	4095522.00	55659507.36
21	2018	7	4266163.00	59925670.36
22	2018	8	4089701.29	64015371.65
23	2018	9	17758.16	64033129.81
24	2018	10	2358.68	64035488.49

12.Calculate the cumulative sales per month for each year.

In [36]:

```
query = """select years, months , payment, sum(payment)
over(order by years, months) cumulative_sales from
(select year(orders.order_purchase_timestamp) as years,
month(orders.order_purchase_timestamp) as months,
round(sum(payments.payment_value),2) as payment from orders join payments
on orders.order_id = payments.order_id
group by years, months order by years, months) as a
"""
cur.execute(query)
```

```
output = cur.fetchall()
df = pd.DataFrame(output)
df
```

Out[36]:

	0	1	2	3
0	2016	9	1008.96	1008.96
1	2016	10	236361.92	237370.88
2	2016	12	78.48	237449.36
3	2017	1	553952.16	791401.52
4	2017	2	1167632.04	1959033.56
5	2017	3	1799454.40	3758487.96
6	2017	4	1671152.12	5429640.08
7	2017	5	2371675.28	7801315.36
8	2017	6	2045105.52	9846420.88
9	2017	7	2369531.68	12215952.56
10	2017	8	2697585.28	14913537.84
11	2017	9	2911049.80	17824587.64
12	2017	10	3118711.52	20943299.16
13	2017	11	4779531.20	25722830.36
14	2017	12	3513605.92	29236436.28
15	2018	1	4460016.72	33696453.00
16	2018	2	3969853.36	37666306.36
17	2018	3	4638608.48	42304914.84
18	2018	4	4643141.92	46948056.76
19	2018	5	4615928.60	51563985.36
20	2018	6	4095522.00	55659507.36
21	2018	7	4266163.00	59925670.36
22	2018	8	4089701.29	64015371.65
23	2018	9	17758.16	64033129.81
24	2018	10	2358.68	64035488.49

13.Calculate the year-over-year growth rate of total sales.

In [39]:

```
query = """with a as(select year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value),2) as payment from orders join payments
on orders.order_id = payments.order_id
group by years order by years)

select years, ((payment - lag(payment, 1) over(order by years))/
lag(payment, 1) over(order by years)) * 100 from a"""
```

```
cur.execute(query)
output = cur.fetchall()
df = pd.DataFrame(output, columns = ["years", "yoy % growth"])
df
```

Out[39]:

	years	yoy % growth
0	2016	NaN
1	2017	12112.703757
2	2018	20.000924

14.Calculate the retention rate of customers, defined as the percentage of customers who make another purchase within 6 months of their first purchase.

In [40]:

```
query = """with a as (select customers.customer_id,
min(orders.order_purchase_timestamp) first_order
from customers join orders
on customers.customer_id = orders.customer_id
group by customers.customer_id),

b as (select a.customer_id, count(distinct orders.order_purchase_timestamp) next_or
from a join orders
on orders.customer_id = a.customer_id
and orders.order_purchase_timestamp > first_order
and orders.order_purchase_timestamp <
date_add(first_order, interval 6 month)
group by a.customer_id)

select 100 * (count( distinct a.customer_id)/ count(distinct b.customer_id))
from a left join b
on a.customer_id = b.customer_id ;"""

cur.execute(query)
output = cur.fetchall()

output
```

Out[40]:

```
[(None,)]
```

15.Identify the top 3 customers who spent the most money in each year.

In [44]:

```
query = """select years, customer_id, payment, d_rank
from
(select year(orders.order_purchase_timestamp) years,
orders.customer_id,
sum(payments.payment_value) payment,
dense_rank() over(partition by year(orders.order_purchase_timestamp)
order by sum(payments.payment_value) desc) d_rank
from orders join payments
on payments.order_id = orders.order_id
group by year(orders.order_purchase_timestamp),
```

```
orders.customer_id) as a
where d_rank <= 3 ;"""

cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["years","id","payment","rank"])
sns.barplot(x = "id", y = "payment", data = df, hue = "years")
plt.xticks(rotation = 90)
plt.show()
```