KwiMart

December 4, 2022

[1]:
```
# Super Market Management System
from mysql.connector import connect, Error
from getpass import getpass
import matplotlib.pyplot as plt
import math
import numpy as np
import pandas as pd
import geopandas as gpd
from termcolor import colored
from IPython.display import display, Image
from datetime import date, datetime

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# for text bold
b_s = "\033[1m" # to start bold characters
b_e = "\033[0;0m" # to end bold characters
```

[3]:
```
# establishing the connection
try:
        conn = connect(
            host='localhost',
            user='root',
            password=getpass(b_s+'Enter password'+b_e),
            database='kwikmart'
        );
        display(Image(filename='Kwik Mart banner.png'))
except Error as e:
        print(e)
```

```
Enter password·····
```

0.1 Working with views, functions and stored procedures

0.1.1 Views

1) Employee Details: This view has only the basic details of all the active employees across all the stores. All the sensetive informations such as salary, SSN number, and higher level employess (CEO's) information has been hidden.

[4]:
```
query='''
    SELECT * FROM kwikmart.employee_details
    ''';
df = pd.read_sql(query, conn)
df
```

[4]:

	emp_id	role	store_id	\
0	1	Cleaner	94	
1	2	Team Lead	78	
2	5	Restock Person	73	
3	6	Packing Person	35	
4	7	Inventory Maintainer	69	
..	…	…	…	
237	488	Cleaner	48	
238	489	County Manager	23	
239	490	Board of Directors	58	
240	495	Cleaner	76	
241	496	Bussiness Head	70	

	store_address	phone	emp_name	\
0	Trantow Passage, West Mateoshire	986-791-7914	Alford Kettridge	
1	Douglas Underpass, Oberbrunnerstad	398-976-3403	Jaye Dimitriev	
2	Schmeler Overpass, Karliton	415-445-1792	Caria Atwater	
3	Sigurd Prairie, New Arne	545-821-2770	Eugenie Barczewski	
4	Marielle Spur, North Dawson	774-382-0966	Brnaba Keattch	
..	…	…	…	
237	Reichel Run, East Horacioview	962-691-1644	Eddie Izhakov	
238	Collier Underpass, Hodkiewiczbury	178-305-9421	Bev Dalbey	
239	Ezra Vista, Nedmouth	437-365-1788	Pedro Rohloff	
240	Sauer Motorway, East Korbinmouth	250-588-7621	Jeanette Sleit	
241	McKenzie Harbor, New Maximomouth	378-773-5917	Theodor Habbema	

```
                emp_address
0            87 Springs Pass
1           52 Spenser Court
2       20 Bobwhite Crossing
3             20 Ohio Avenue
4            347 Veith Court
..                       …
237           3 Knutson Pass
238          1788 Ryan Plaza
239       92497 Elka Terrace
240    31330 Bobwhite Avenue
241     592 Thackeray Circle

[242 rows x 7 columns]
```

2) Current month's store revenue: This view helps the upper management to track the current month's revenue along with the previous day revenue across all the stores.

[5]:
```
query='''
    SELECT * FROM kwikmart.current_month_store_revenue
    ''';
df = pd.read_sql(query, conn)
df
```

[5]:

	store_id	store_detail	store_email	\
0	20	Nikko Station , Kemmerfort	93agf@kwikmart.com	
1	48	Reichel Run , East Horacioview	07brx@kwikmart.com	
2	42	Zulauf Locks , Lake Dockbury	56gtb@kwikmart.com	
3	16	Henderson Vista , East Tracefort	12lwm@kwikmart.com	
4	94	Trantow Passage , West Mateoshire	73cjj@kwikmart.com	
5	9	Davin Village , New Nova	71zrl@kwikmart.com	
6	21	D'Amore Pine , Stammhaven	15maw@kwikmart.com	
7	31	Tremblay Valley , West Dellview	90dtb@kwikmart.com	
8	1	Kiehn Well , Romabury	40nfv@kwikmart.com	
9	95	Vicenta Springs , Prohaskahaven	88ssu@kwikmart.com	
10	27	Weimann Branch , East Cristinaburgh	21zqc@kwikmart.com	
11	74	Lebsack Drive , New Jaqueline	46yyd@kwikmart.com	
12	56	Adams Trail , Schummtown	91rkd@kwikmart.com	
13	84	Kiehn Mill , Gleichnerport	07wjr@kwikmart.com	
14	50	Missouri Lake , Onamouth	36qvx@kwikmart.com	
15	67	King Manors , Davisland	48pha@kwikmart.com	
16	76	Sauer Motorway , East Korbinmouth	46iba@kwikmart.com	
17	60	Neva Trail , Lake Elna	22hqq@kwikmart.com	
18	19	Dante Islands , Marquardtburgh	10jna@kwikmart.com	
19	34	Demarco Mount , Aufderharborough	14ltm@kwikmart.com	
20	65	Sipes Valleys , East Queen	38rgi@kwikmart.com	
21	14	Branson Extension , Scotburgh	80agk@kwikmart.com	

```
22        82           Jones Port , North Rosalia  64nny@kwikmart.com
23        63       Raquel Lakes , Lake Alethafurt  36wlw@kwikmart.com
24        68         Jonathon Parkway , Reneefurt  12nvq@kwikmart.com
```

	net_revenue_till_date	previous_day_revenue	total_bills_till_date
0	755.21	0.0	2
1	606.19	0.0	2
2	579.90	0.0	1
3	474.38	0.0	2
4	415.92	0.0	1
5	316.27	0.0	1
6	273.65	0.0	1
7	270.23	0.0	1
8	233.01	0.0	2
9	217.61	0.0	1
10	184.42	0.0	1
11	184.30	0.0	1
12	180.78	0.0	1
13	150.78	0.0	1
14	134.80	0.0	1
15	114.52	0.0	1
16	106.95	0.0	1
17	99.40	0.0	1
18	90.16	0.0	1
19	86.25	0.0	1
20	76.06	0.0	1
21	72.37	0.0	1
22	45.00	0.0	1
23	12.55	0.0	1
24	5.72	0.0	1

0.1.2 Functions

1) Net revenue for a particular store in a given date range: This helps each store track their performance across months or days.

[6]:
```
cursor = conn.cursor()
```

[7]:
```
func = "SELECT store_billing(%s, %s, %s)"
store_id = input(b_s+"Please enter store id: ")
start_date = input(b_s+"From date: ")
end_date = input(b_s+"Till date: ")

result = cursor.execute(func, [store_id, start_date, end_date])
print(b_s+f"Total revenue for store id {store_id} between {start_date} and␣
 ↪{end_date} is: "+b_e, cursor.fetchone()[0],)

# 1 | 2022-01-01 | 2022-12-31
```

```
Please enter store id: 1
From date: 2022-01-01
Till date: 2022-12-31
Total revenue for store id 1 between 2022-01-01 and 2022-12-31 is:
1086.64
```

2) Update discount for product: This function let store manager create new discount or update the previous discounts along with an option to make it active or inactive

[8]:
```
# item_ids :␣
 ↪5,8,12,58,99,127,147,169,181,205,211,213,216,235,261,267,282,285,302
```

[9]:
```
store_id = input(b_s+"Please enter store id: ")
item_id = input(b_s+"Please enter item id: ")

query='''
    SELECT * FROM kwikmart.discount
    WHERE store_id = %s AND item_id = %s
    ''';
df = pd.read_sql(query, conn, params= [store_id, item_id])

if df.empty:
    print(b_s+f"The discount for item id {item_id} does not exist. Please add␣
 ↪the discount in next step. ")
else:
    display(df)

#1 1 | 285
```

```
Please enter store id: 1
Please enter item id: 99
The discount for item id 99 does not exist. Please add the discount in next
step.
```

[10]:
```
func = "SELECT update_discount(%s, %s, %s, %s)"
percent_off = input(b_s+"Please enter discount %: ")
is_active = input(b_s+"Please enter 1 to make discount live and 0 to remove␣
 ↪discount: ")

result = cursor.execute(func, [store_id, item_id, percent_off, is_active])
print(cursor.fetchone())

df = pd.read_sql(query, conn, params= [store_id, item_id])
conn.commit()
df
```

```
Please enter discount %: 10
Please enter 1 to make discount live and 0 to remove discount: 1
(1,)
```

[10]:

	store_id	item_id	percent_off	is_active
0	1	99	10.0	1

3) Net revenue for all stores in a given date range: This helps track their performance across months or days.

[11]:
```
func = "SELECT overall_billing(%s, %s)"
start_date = input(b_s+"From date: ")
end_date = input(b_s+"Till date: ")

result = cursor.execute(func, [start_date, end_date])
print(b_s+f"Total revenue for all stores between {start_date} and {end_date} is:
  ↪ "+b_e, cursor.fetchone()[0],)

# 2022-01-01 | 2022-12-31
```

```
From date: 2022-01-01
Till date: 2022-12-31
Total revenue for all stores between 2022-01-01 and 2022-12-31 is:
38775
```

4) Create new bill: This function initiates a new bill and generate a bill_id in bill table.

[12]:
```
#create bill
func = "SELECT create_bill(%s, %s, %s)"
store_id = input(b_s+"Please enter store id: ")
phone_no = input(b_s+"Please enter phone number: ")
dt = date.today()
date_ = dt.strftime("%Y-%m-%d")

result = cursor.execute(func, [store_id, phone_no, date_])
bill_id = cursor.fetchone()[0]
print(b_s+f"Bill id {bill_id} generated. ")

conn.commit()
# 1 | 100-736-5070
```

```
Please enter store id: 1
Please enter phone number:
Bill id 196 generated.
```

0.1.3 Stored procedures

Adding item to cart: This stored procedure let customers add one or more items to cart.

[13]:
```
def AddToCart(conn,bill_id,before_order):
    item = np.zeros(3,dtype=np.int)

    item_id = input(b_s+"Please enter item_id: ")
```

```
    quantity = input(b_s+"Please enter quantity: ")

    item[0] = bill_id
    item[1] = item_id
    item[2] = quantity

    qry = '''
    SELECT si.store_id, si.item_id, si.qty_in_stock FROM store_item si
    WHERE si.item_id = %s
    AND si.store_id = (SELECT b.store_id FROM bill b WHERE b.bill_id = %s)
    '''
    df = pd.read_sql(qry, conn, params=[item_id, bill_id])
    before_order = pd.concat([before_order, df], ignore_index=True)

    cursor.callproc('add_item_to_cart', item.tolist())
    conn.commit()

    return before_order

before_order = pd.DataFrame(columns = ['store_id', 'item_id', 'qty_in_stock'])
resp = 'Yes'
```

[14]:
```
# item_ids :␣
↪5,8,12,58,99,127,147,169,181,205,211,213,216,235,261,267,282,285,302
```

[15]:
```
# add item to bill

while resp == 'Yes':
    before_order = AddToCart(conn,bill_id,before_order)
    resp = input(f"Do you want to add more item Yes|No: ")
```

```
Please enter item_id: 5
Please enter quantity: 1
Do you want to add more item Yes|No: Yes
Please enter item_id: 58
Please enter quantity: 2
Do you want to add more item Yes|No: Yes
Please enter item_id: 99
Please enter quantity: 2
Do you want to add more item Yes|No: Yes
Please enter item_id: 205
Please enter quantity: 3
Do you want to add more item Yes|No: Yes
Please enter item_id: 302
Please enter quantity: 1
Do you want to add more item Yes|No: No
```

[16]:
```
#resp = 'Yes'
```

[17]:
```
print(b_s+"List of items and the Quantity in Stock before item billing.")
before_order
```

List of items and the Quantity in Stock before item billing.

[17]:

	store_id	item_id	qty_in_stock
0	1	5	94
1	1	58	143
2	1	99	78
3	1	205	82
4	1	302	173

[18]:
```
qry = '''
    SELECT si.store_id, si.item_id, si.qty_in_stock FROM bill b
    JOIN bill_items bi ON bi.bill_id = b.bill_id
    JOIN store_item si ON si.store_id = b.store_id AND si.item_id = bi.item_id
    WHERE b.bill_id = %s
    '''
after_bill = pd.read_sql(qry, conn, params=[bill_id])
print(b_s+"List of items and the Quantity in Stock after item billing.")
after_bill
```

List of items and the Quantity in Stock after item billing.

[18]:

	store_id	item_id	qty_in_stock
0	1	5	93
1	1	58	141
2	1	99	76
3	1	205	79
4	1	302	172

The above result helps us to visualize the working of Trigger that we have setup on 'bill_items' table.

2) Generating the bill for customer: Using this stored procedure, we can print the receipt if a customer wants to. This receipt include all the details like Store address, billing time, Items, quantity, price and the discount offered.

[19]:
```
query=f'''
    SELECT bill.store_id from bill where bill.bill_id = {bill_id}
    ''';
df = pd.read_sql(query, conn)
store_id = df.store_id[0]

query=f'''
    SELECT CONCAT(store.street_name,", ",store.city,", ",store.zip_code)␣
 ↪address from store where store_id = {store_id}
    ''';
df = pd.read_sql(query, conn)
```

```
store_addr = df.address[0]

cursor.callproc('get_bill_amount', [bill_id])
get_bill_amount_results = cursor.stored_results()

print(b_s+f'%15s %30s %15s' % ('', 'KwikMart Supermarket', ''))
print(b_s+f'%18s %30s %15s' % ('', store_addr, ''))
print(b_s+f'%18s %30s %15s' % ('', datetime.today(), ''))
print(b_s+f'%14s %30s %10s' % ('', f'Bill Number: {bill_id}', ''))
print()

idx = 0;
table_content = []
total = 0;
net = 0;
total_discount = 0
print('%30s  %12s  %12s' % ('Item', 'Quantity', 'Amount'))
print(' ' * 60)
for r in get_bill_amount_results:
    if (idx == 0):
        for value in r.fetchall():
            item_name = value[0];
            quantity = value[1];
            unit_price = value[2];
            item_discount = value[3];
            item_net = value[4];
            line = '%30s  %12s  %12s' % (item_name, quantity, unit_price)
            print(line)
            if (item_discount != 0) :
                print('%45s %12s' % ('Discount: ', item_discount))
        print(' ' * 60)

    if (idx == 1):
        for value in r.fetchall():
            net = value[0]
    if (idx == 2):
        for value in r.fetchall():
            total_discount = value[0]
    if (idx == 3):
        for value in r.fetchall():
            total = value[0]
    idx = idx + 1;

print('%45s %12s' % ('Total', total))
print('%45s %12s' % ('Discount', total_discount))
print('%45s %12s' % ('Net', net))
```

```
                    KwikMart Supermarket
                  Kiehn Well, Romabury, 78377
                   2022-12-04 16:19:38.349648
                        Bill Number: 196
                    Item      Quantity        Amount

      Vitamins / Supplements             1         23.11
                            Discount:            3.47
           Insect repellent             2         23.88
                    Vinegar             2         21.95
                            Discount:            2.20
               Buns / Rolls             3         17.69
               Burger night             1         17.00

                                    Total       184.84
                                 Discount         7.87
                                      Net       176.99
```

Item availability at other stores: This stored procedure helps store person to recommend other nearby stores to the customer for any particular item. This helps customer to procure the items from nearby stores for unavailable items.

[20]:
```
# item_ids :␣
↪5,8,12,58,99,127,147,169,181,205,211,213,216,235,261,267,282,285,302
```

[21]:
```
store_id = input(b_s+"Please enter Store Id: ")
item_id = input(b_s+"Please enter Item Id: ")

cursor.callproc('item_store_check', [store_id, item_id,])
results = cursor.stored_results()

for r in results:
    for value in r.fetchall():
        addr = value[3]
        quantity = value[2]
        print(b_s+f"Store Address: {addr} Quantity: {quantity} ")
```

```
Please enter Store Id: 1
Please enter Item Id: 100
```

```
Store Address: Erdman Road, East Lorainefort, 68136 Quantity: 36

Store Address: Sauer Motorway, East Korbinmouth, 56259 Quantity: 133

Store Address: Jaycee Glens, Lake Jaydeland, 17644 Quantity: 17
```

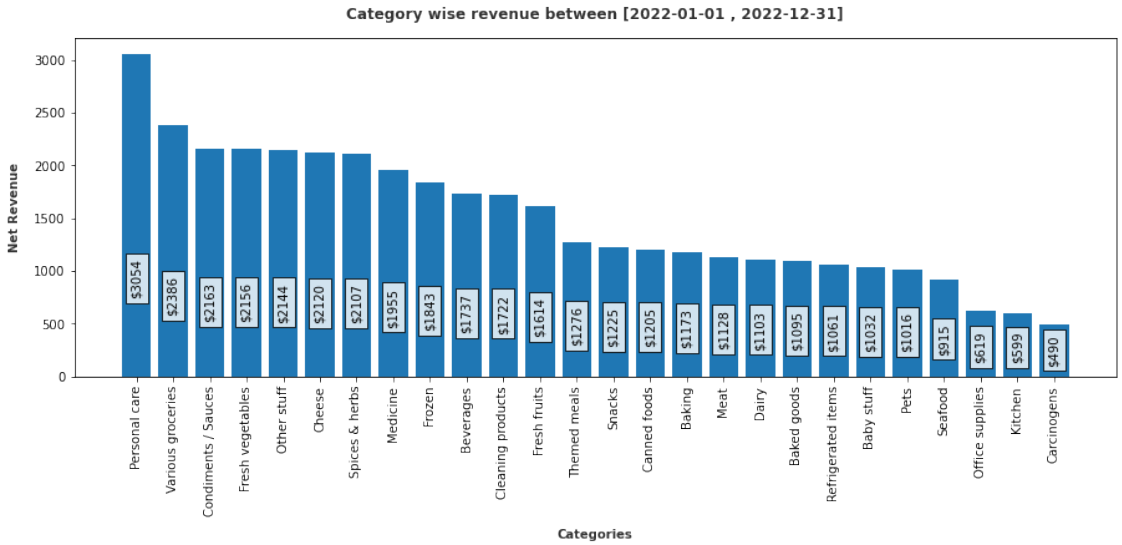
0.2 Some data trends at Kwik Mart

0.2.1 Looking at the category wise revenue across all stores

[22]:
```
def getCategoriesRevenue(conn, start_date, end_date):
    query='''
    SELECT c.category_name AS Category_Name ,SUM(bi.quantity * bi.net_price) AS␣
 ↪Revenue
    FROM bill b
    JOIN bill_items bi ON bi.bill_id = b.bill_id
        AND b.bill_date >= %s and b.bill_date <= %s
    JOIN item i ON i.item_id = bi.item_id
    JOIN category c ON c.category_id = i.category_id
    GROUP BY Category_Name
    ORDER BY Revenue DESC
    ''';

    # function to add value labels
    def addlabels(x,y):
        for i in range(len(x)):
            amt = '$'+str(math.trunc(y[i]))
            plt.text(i, y[i]//4, amt, ha = 'center',
                     bbox = dict(facecolor = 'white', alpha =.8),
                    rotation = 90)

    value_tuple=(start_date, end_date);
    df = pd.read_sql(query, conn, params=[start_date, end_date])
    plt.figure(figsize=(15, 5))
    plt.bar(df.Category_Name, df.Revenue)
    plt.xticks(rotation=90)

    addlabels(df.Category_Name, df.Revenue)

    # Add labels and a title.
    plt.xlabel('Categories', labelpad=10, color='#333333',weight='bold')
    plt.ylabel('Net Revenue', labelpad=10, color='#333333',weight='bold')
    plt.title(f'Category wise revenue between [{start_date} , {end_date}]',␣
 ↪pad=15, color='#333333',weight='bold')

    plt.show()
```

[23]:
```
# calling getCategoriesRevenue function
print(b_s+"Category wise sales across all stores, please enter the date range:
 ↪"+b_e)
cat_rev_start = input(b_s+"From date: "+b_e)
cat_rev_end = input(b_s+"Till date: "+b_e)
getCategoriesRevenue(conn, cat_rev_start, cat_rev_end)
#2022-01-01 | 2022-12-31
```

Category wise sales across all stores, please enter the date range:
From date: 2022-01-01
Till date: 2022-12-31



0.2.2 State wise no of stores and revenue

[24]:
```
def numOfstoresVsRevenue(conn):
    query1 = '''
    Select t1.state, t1.numOfStores, t2.revenue from
    (select count(*) as numOfStores, store.state
    from store
    group by store.state
    order by store.state) t1 join
    (    select store.state, SUM(revenue) as revenue
    from store left outer join (
    select bill.store_id, Sum(bill_items.quantity * bill_items.net_price) as␣
 ↪revenue
    from bill, bill_items
    where bill.bill_id = bill_items.bill_id
    group by bill.store_id) temp
    on store.store_id = temp.store_id
```

```
    group by store.state) t2
    where t2.state = t1.state
    order by t1.numOfStores
    ''';

    df = pd.read_sql(query1, conn);
    return df
```

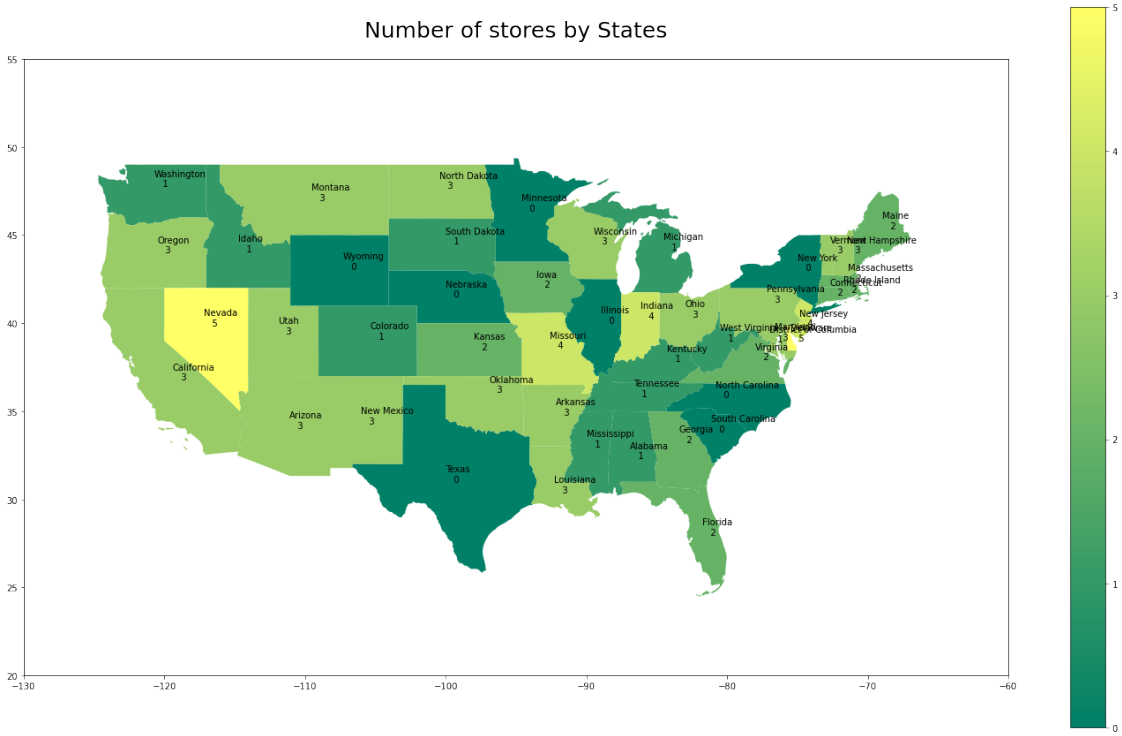
[25]:
```
# calling function
df = numOfstoresVsRevenue(conn)
df.rename(columns={"state": "NAME"},inplace=True)
di = {"SouthDakota":"South Dakota", "WestVirginia":"West Virginia",␣
 ↪"RhodeIsland":"Rhode Island", "NewHampshire":"New Hampshire",
               "NewMexico":"New Mexico", "NorthDakota": "North␣
 ↪Dakota","NewJersey":"New Jersey"}
df = df.replace({"NAME": di})

# getting latitude and longitude for each state
df_lat_long = pd.read_csv("US_States_lat_lon_final.csv")
df_states = df_lat_long.merge(df,on='NAME',how='outer')
df_states[["numOfStores","revenue"]] = df_states[["numOfStores","revenue"]].
 ↪fillna(0)
df_states.head()
```

[25]:

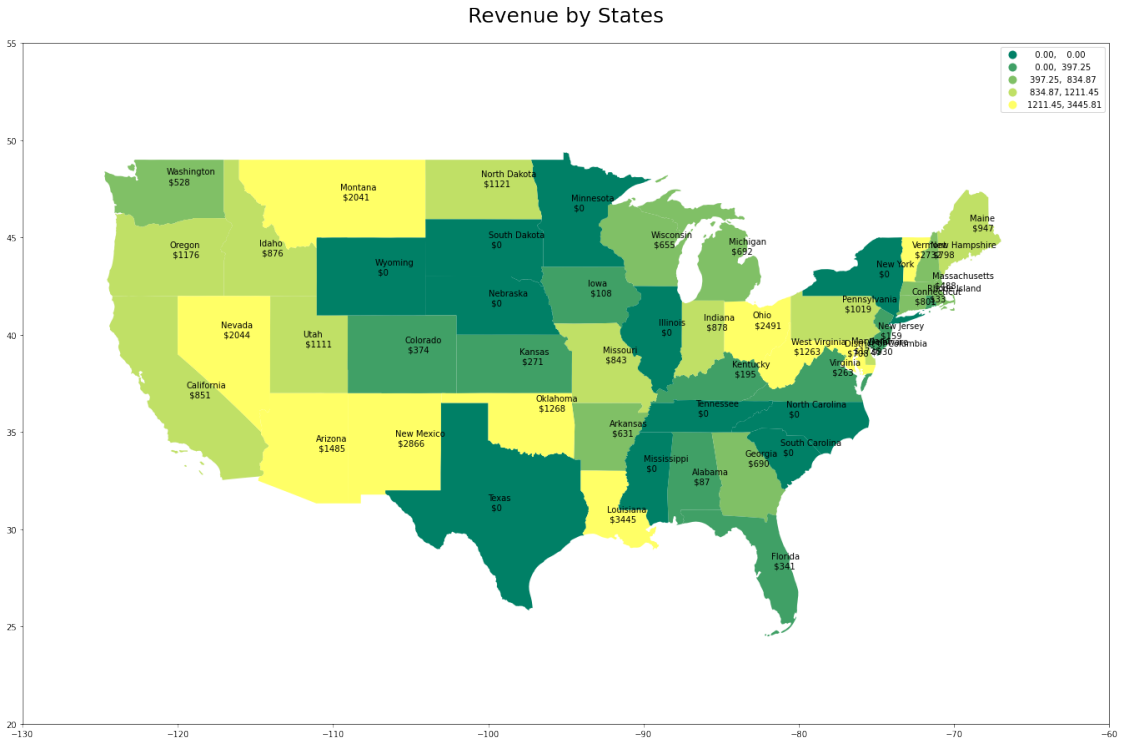
	NAME	lat	lon	numOfStores	revenue
0	Maryland	39.045753	-76.641273	3.0	1726.23
1	Iowa	42.032974	-93.581543	2.0	108.63
2	Delaware	39.000000	-75.500000	5.0	930.27
3	Ohio	40.367474	-82.996216	3.0	2491.55
4	Pennsylvania	41.203323	-77.194527	3.0	1019.02

[26]:
```
# url of our shape file
path=r"C:\Users\agraw\Documents\Git\DS5110\Python␣
 ↪implementation\cb_2018_us_state_20m"
# load the shape file using geopandas
geo_usa = gpd.read_file(path+'\cb_2018_us_state_20m.shp')

# merge usa_state data and geo_usa shapefile
geo_merge=geo_usa.merge(df_states,on='NAME',how='outer')

indexRow = geo_merge[ (geo_merge.NAME == 'Alaska') | (geo_merge.NAME ==␣
 ↪'Hawaii') | (geo_merge.NAME == 'Puerto Rico')].index
geo_merge.drop(indexRow , inplace=True)
geo_merge.reset_index(drop=True, inplace=True)

# plot state wise number of stores
geo_merge.plot(column='numOfStores', figsize=(25, 15),legend=True,cmap='summer')
```

```
plt.xlim(-130,-60)
plt.ylim(20,55)
# add countries names and numbers
for i in range(len(geo_merge)):
    plt.text(geo_merge.lon[i],geo_merge.lat[i],"{}\n   {}".format(geo_merge.
 ↪NAME[i],int(geo_merge.numOfStores[i])),size=10)
plt.title('Number of stores by States',pad=25,fontsize=25)
plt.show()
```



[27]:
```
# plot state wise revenue
geo_merge.plot(column='revenue',scheme="quantiles", figsize=(25,␣
 ↪15),legend=True,cmap='summer')
plt.xlim(-130,-60)
plt.ylim(20,55)
# add countries names and numbers
for i in range(len(geo_merge)):
    plt.text(geo_merge.lon[i],geo_merge.lat[i],\
             "\n{}\n ${}".format(geo_merge.NAME[i],int(geo_merge.
 ↪revenue[i])),size=10)
plt.title('Revenue by States',pad=25,fontsize=25)
plt.show()
```



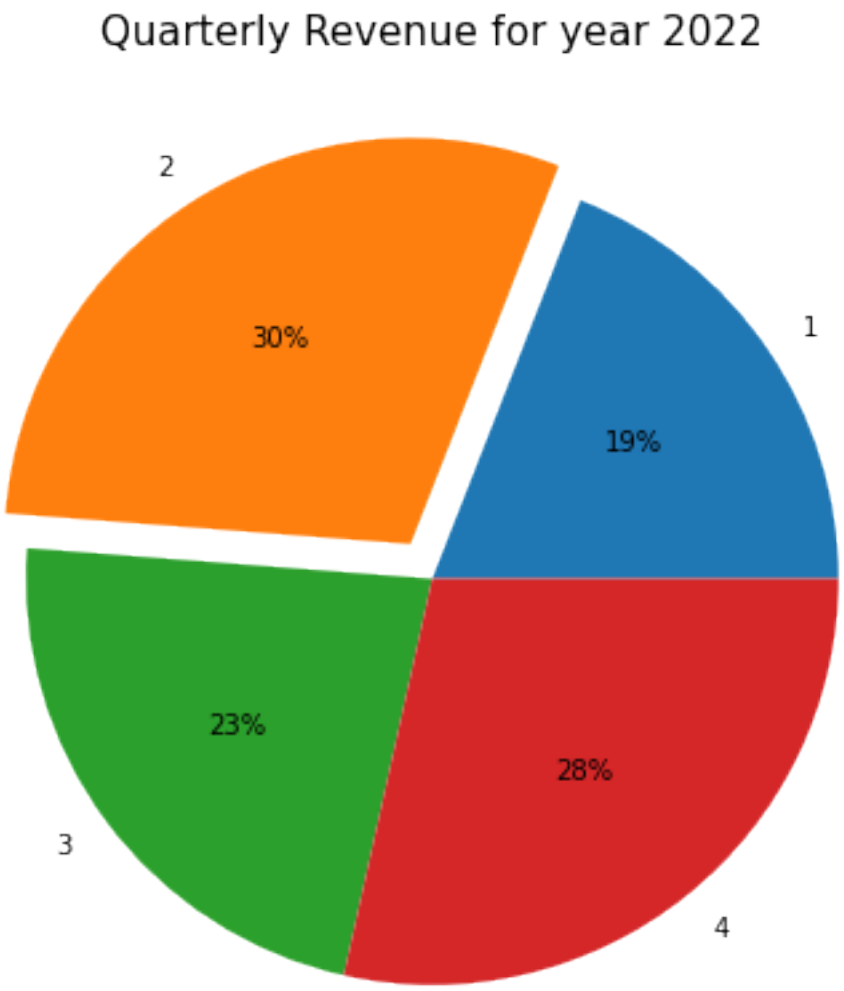
0.2.3 Quarter-wise business performance based on Revenue

[28]:
```
def revenueInEachQuarter(conn, year):
    query='''
    SELECT CEILING(MONTH(b.bill_date)/3) AS quarter, SUM(bi.net_price*bi.
 ↪quantity) net_revenue
    FROM bill_items bi
    JOIN bill b ON b.bill_id = bi.bill_id
    WHERE b.bill_date LIKE %s
    GROUP BY 1
    ORDER BY quarter
    ''';

    value_tuple = (year+'%',)
    df = pd.read_sql(query, conn,params = value_tuple)
    idx = df[['net_revenue']].idxmax()
    myexplode = np.zeros(len(df.quarter))
    myexplode[idx[0]] = 0.1

    plt.figure(figsize=(7, 7))
    plt.pie(df.net_revenue, labels = df.quarter,autopct='%.0f%%', explode =␣
 ↪myexplode)
    plt.title(f'Quarterly Revenue for year {year}',pad=10,fontsize=15)
```

```
    plt.show()
```

[29]:
```
# pie-chart to visulize quarterly performance
qtr_year = input(b_s+"Quarterly performance, please enter the year: "+b_e)
revenueInEachQuarter(conn,qtr_year)
```

```
Quarterly performance, please enter the year: 2022
```



[30]:
```
conn.close()
cursor.close()
```

[30]:
```
True
```