US Based Retail Store Sales Analysis

INTCDB22DW075

BATCH - 1

Contents

1. Business Challenge / Requirement

An US based Retail Store is facing challenges to analysis the data as they have the data present in different sources and understanding their customers show of interest on buying their products. So, the company wants to take the help of Data Analytics using Big Data technologies to analyse the data of users in order to increase its sales, profit and also prevent from big losses by investing money on unsold products.

2. The goal of the project

The goal of the project is to prepare a sales report that helps the company to improve its profit. Providing the insights that are extracted from the data, show case the sales report as a graph format.

3. Project Architecture

1. The company shared 5 main files that contains the format as .csv, .txt and table with name as Customer, Sales, Product Review, Product Name and Branch respectively.

2. The file data and tables are validated, enriched, and processed before loading into HDFS/HIVE.

3. After analysing queries, applying transformation logic, we created the visualisation charts which helps the company to understand the trends and pattern of sales in order to increase their profit.

US Based Retail Store Analysis Architecture

Uploading data to LINUX

- Customer.csv (3 files)
- sales.csv (3 files)
- Branch(table)
- Product review.csv
- Product.txt

Data Cleaning

- Customer.csv (3 files)
- sales.csv (3 files)
- Branch(table)
- Product review.csv
- Product.txt

Data Integration

- Customer.csv
- sales.csv
- Branch(table)
- Product review.csv
- Product.txt

Hadoop

Data Ingestion Services

HIVE , SQOOP

Raw Data → Preprocessed Data → Matplotlib, Seaborn

Data Storage

hadoop HDFS

Data Preprocessing Framework

python Programming

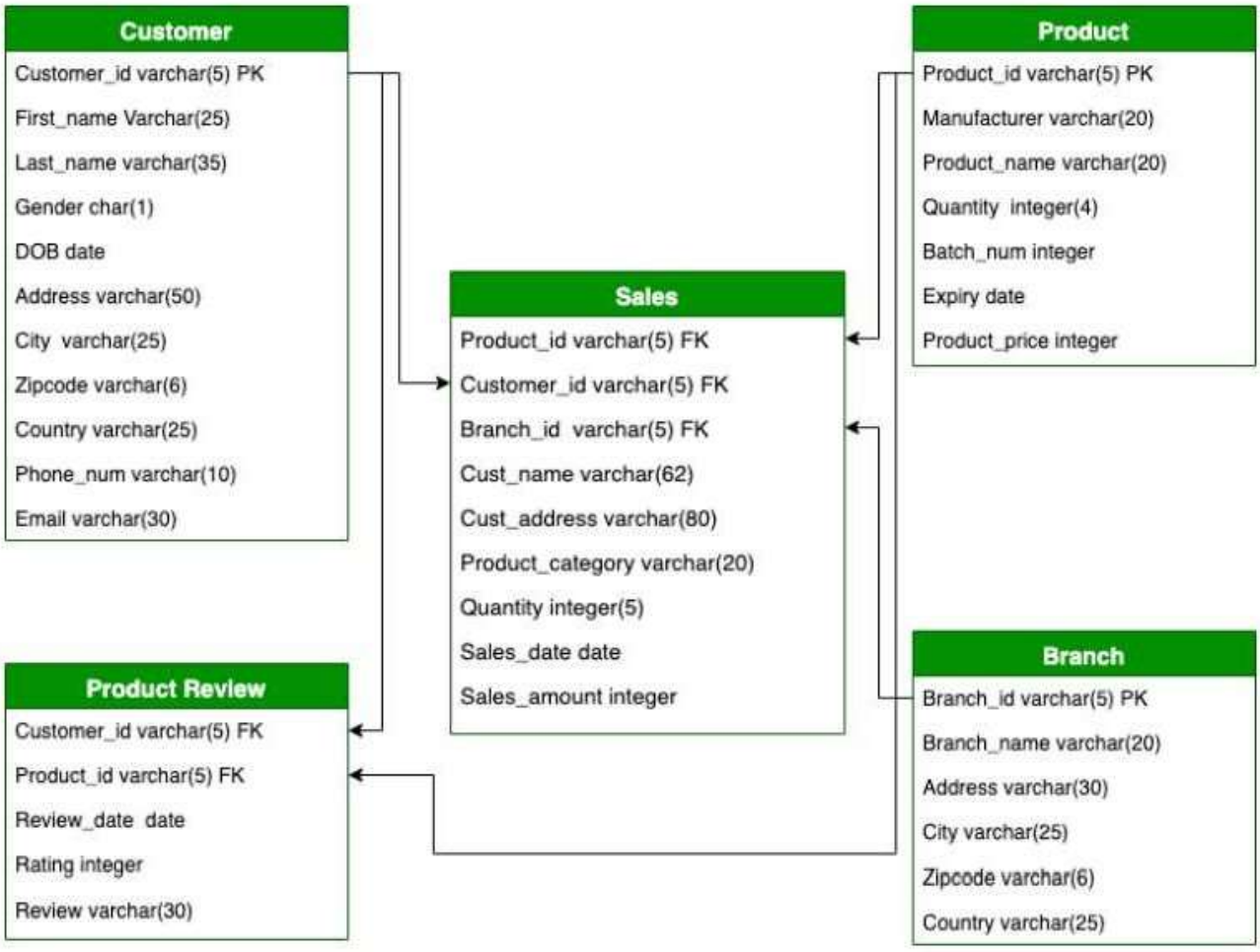
Consumer Services

Insights From the Data

4. Dataset Explanation and Schema

1. Data coming from third-party sources reside in the local directory and has a csv, a txt format and a Spreadsheet.
2. A master table is created from the existing tables which is then inserted in HDFS using Sqoop import.
3. Further the Dataset is Visualised using the python Libraries. Matplotlib tool is used to map the dataset.

DATA MODEL



Fields present in the data files and tables

Data files contain below fields

Column Name/Field Name Column Description/Field Description

Customer
CSV File Fields

• **Customer_id**	Unique identifier for customer	Varchar [5]
• **First_name**	First name of the customer	Varchar [25]
• **Last_name**	Last name of the customer	Varchar [35]
• **Gender**	Gender of the customer	Char [1]
• **DOB**	Date of birth of the customer	Date
• **Address**	Location of the customer	Varchar [50]
• **City**	City of the customer	Varchar [25]
• **Zipcode**	Pincode of the address	Varchar [6]
• **Country**	Country of the customer	Varchar [25]
• **Phone_num**	Contact detail of the customer	Varchar [10]
• **Email**	Mail address of the customer	Varchar [30]

Product
Txt File Fields

• **Product_id**	Unique id of the product	**Varchar [5]**
• **Manufacturer**	Name of the manufacturer for the product	**Varchar[20]**
• **Product_name**	Name of the product	**Varchar [20]**
• **Quantity**	No of products ordered	**Varchar [20]**
• **Batch_num**	Unique identifier for batches of the product	**Integer [4]**
• **Expiry_date**	The date after the product may no longer to use	**Date**
• **Product_price**	Price of the product	**Integer**

Product Review
Csv File Fields

• **Customer_id**	Unique identifier of the customer	**Varchar [5]**
• **Product_id**	Unique identifier of the product	**Varchar [5]**
• **Review_date**	Date of review of the product	**Date**
• **Rating**	Points for product review	**Integer**
• **Review**	Review of the product	**Varchar [30]**

Branch
Csv File Fields

• **Branch_id**	Unique identifier of the branch	**Varchar [5]**
• **Branch_name**	Name of the branch	**Varchar [20]**
• **Address**	Location of the branch	**Varchar [30]**
• **City**	City where the branch is located	**Varchar [25]**
• **Zipcode**	Pin code of the city	**Varchar [6]**
• **Country**	Country where the branch is located	**Varchar [25]**

Sales
Csv File Fields

• **Product_id**	Unique identifier of the product	**Varchar [5]**
• **Customer_id**	Unique identifier of the customer	**Varchar [5]**
• **Branch_id**	Unique identifier of the branch	**Varchar [5]**
• **Cust_name**	Name of the customer	**Varchar [62]**
• **Cust_address**	Location of the customer	**Varchar [80]**
• **Product_category**	Category of the product	**Varchar [20]**
• **Quantity**	Number of product order	**Integer [5]**
• **Sales_Date**	Date of the sales	**Date**
• **Sales_amount**	Price of the sales done by customer	**Integer**

Fields present in the target data files and tables

Target Column Name/Field Name Column Description/Field Description

Cust_transformation
CSV File Fields

• **Customer_id**	Unique identifier for customer	**Varchar [5]**
• **Name**	Name of the customer	**Varchar [62]**
• **Age**	Age of the customer	**Integer [2]**
• **Contact**	Customer email and mobile details	**Varchar [10]**
• **Address**	Address, city code and name of the city	**Varchar [60]**
• **State**	Name of the state	**Varchar [25]**

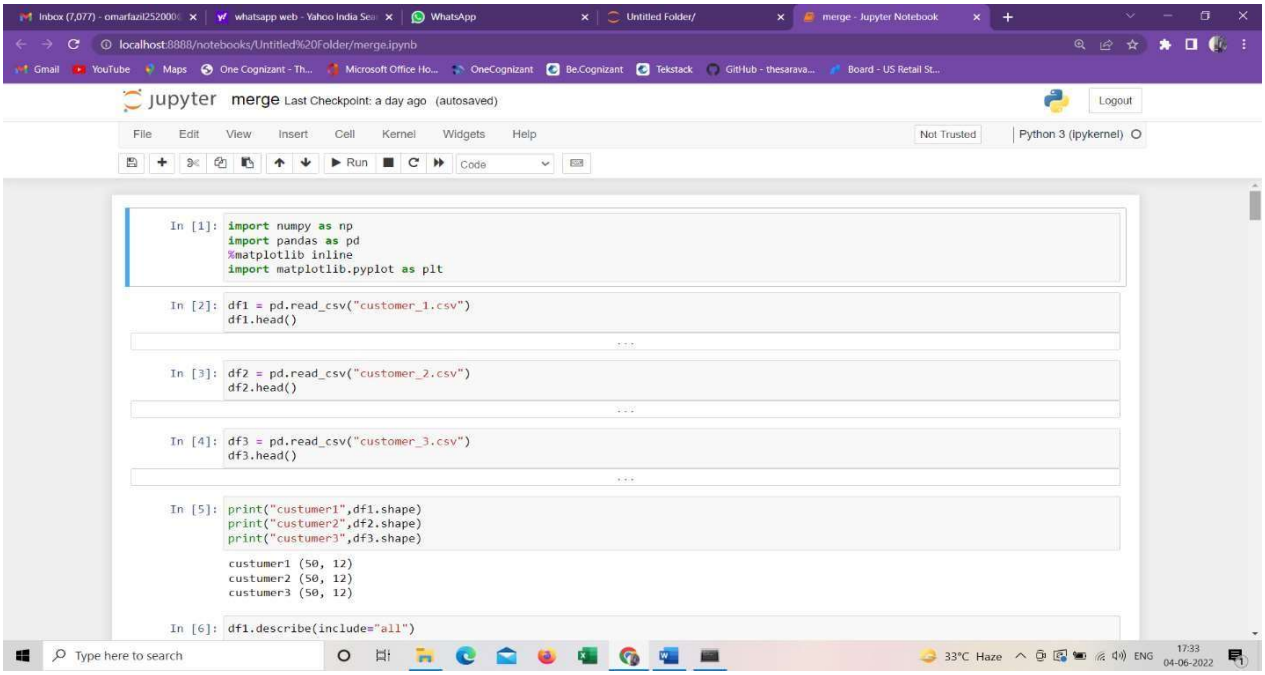
Sales_transformation
CSV File Fields

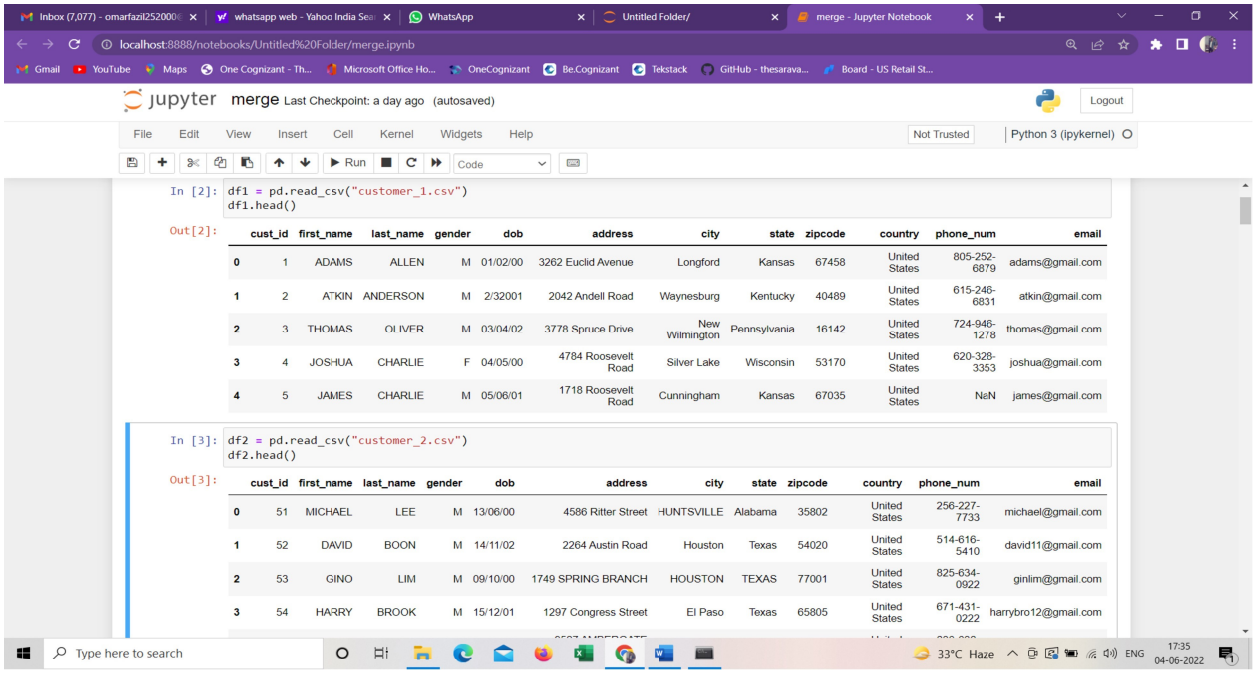
• **Product_id**	Unique identifier of the product	**Varchar [5]**
• **Sales_id**	Unique identifier of the sales	**Varchar [5]**
• **Customer_id**	Unique identifier of the customer	**Varchar [5]**
• **Branch_id**	Unique identifier of the branch	**Varchar [5]**
• **Branch_details**	Details of the branches	**Varchar [35]**
• **Product_detail**	Details of the products	**Varchar [25]**
• **Quantity**	Number of product order	**Integer [5]**
• **Sales_Date**	Date of the sales	**Date**
• **Sales_amount**	Price of the sales done by customer	**Integer**

5. Code Templates

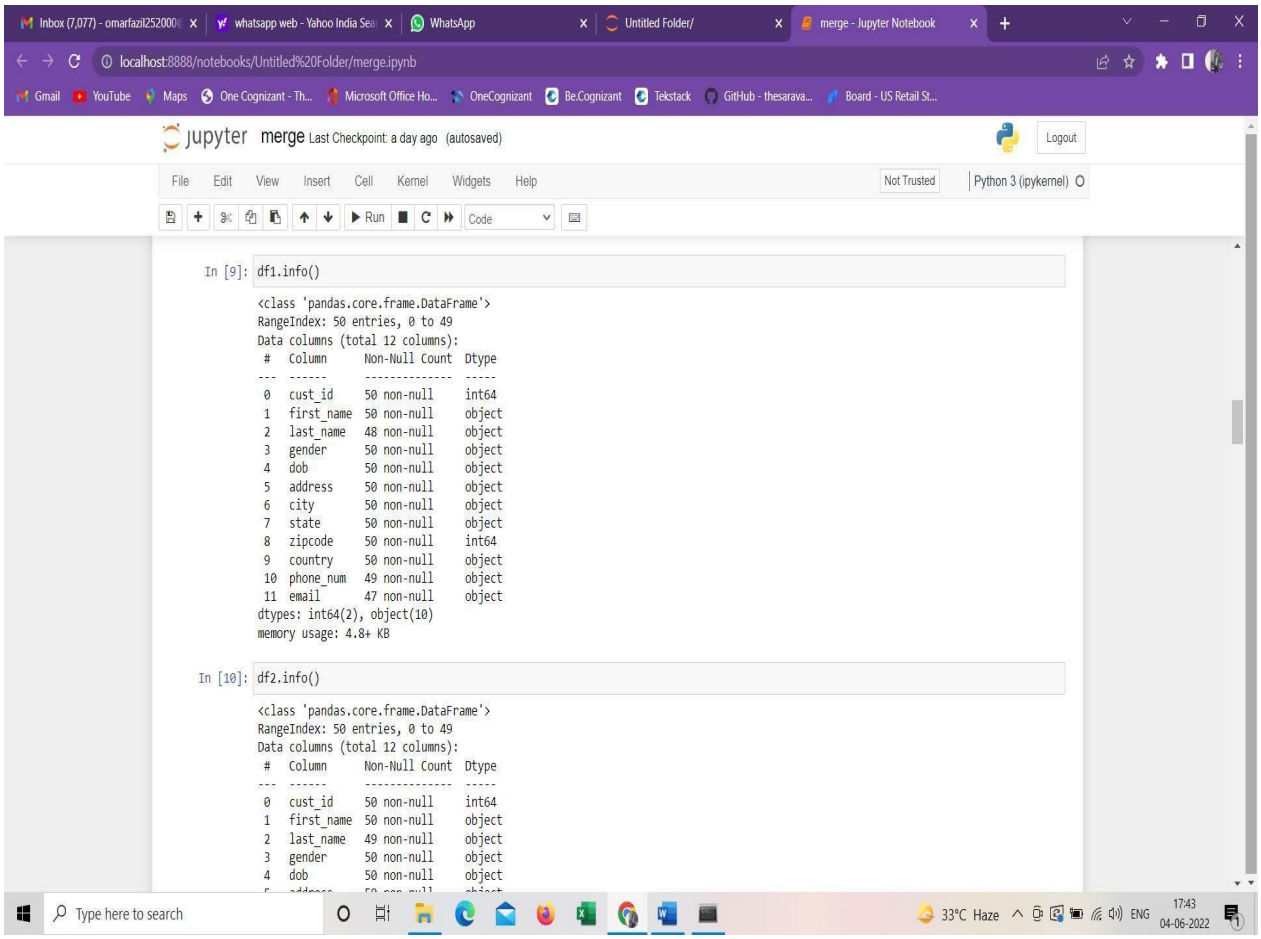
5.1 Data Processing

1. Importing all the required libraries for data pre-processing.
2. Importing the raw data from the source file customer.csv and printing the head of the data.





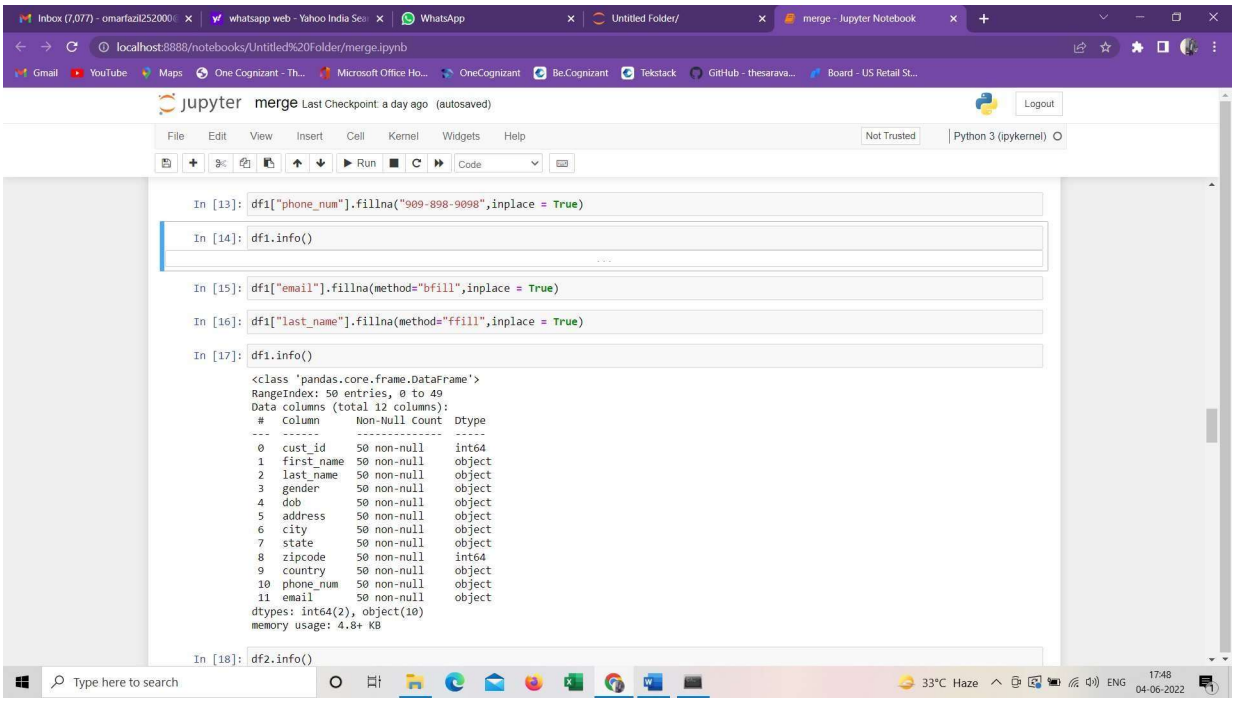
3. Checking null values present in the data



5.1.1 Conversion of raw data to processed data

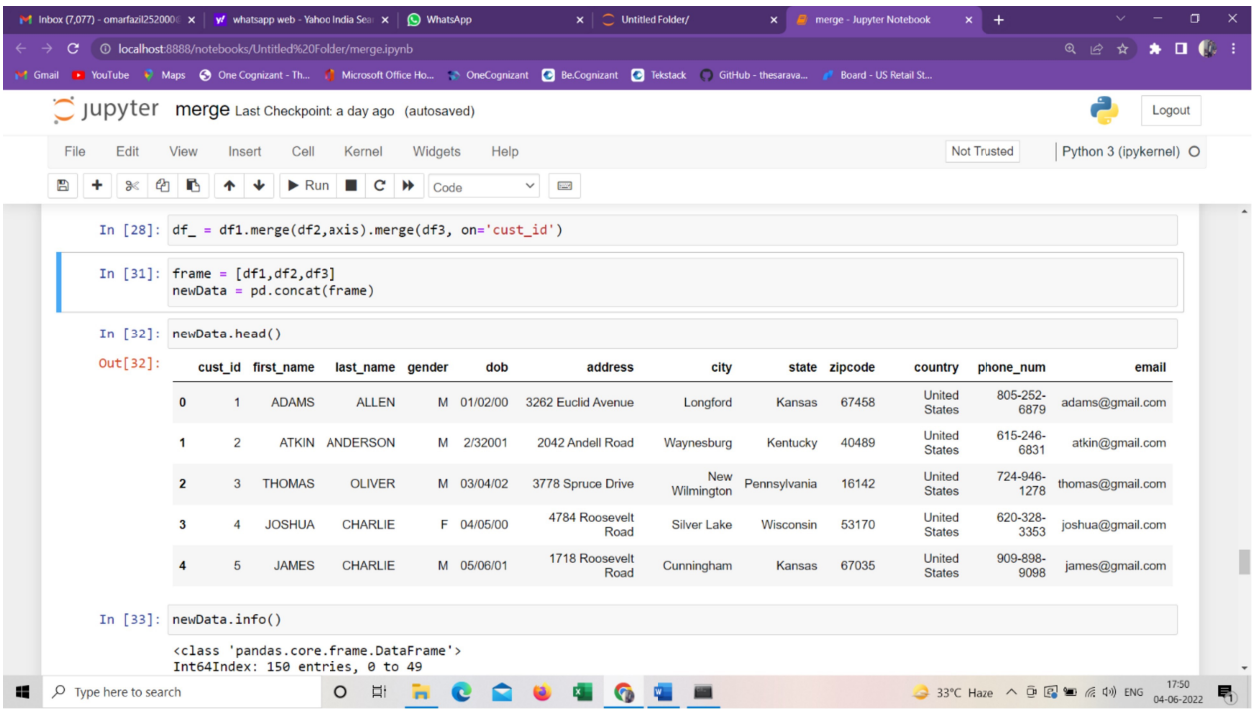
1. For each raw file we have checked null values, duplicate values and other parameters and then converted them into processed dataset. Here are some samples of code.

2. For example the below picture shows the null values present in the customer1.csv file



```
In [13]: df1["phone_num"].fillna("909-898-9098",inplace = True)

In [14]: df1.info()

In [15]: df1["email"].fillna(method="bfill",inplace = True)

In [16]: df1["last_name"].fillna(method="ffill",inplace = True)

In [17]: df1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   cust_id     50 non-null     int64
 1   first_name  50 non-null     object
 2   last_name   50 non-null     object
 3   gender      50 non-null     object
 4   dob         50 non-null     object
 5   address     50 non-null     object
 6   city        50 non-null     object
 7   state       50 non-null     object
 8   zipcode     50 non-null     int64
 9   country     50 non-null     object
 10  phone_num   50 non-null     object
 11  email       50 non-null     object
dtypes: int64(2), object(10)
memory usage: 4.8+ KB

In [18]: df2.info()
```

3. Merge the three df1, df2 and df3 file as a single file named as newData and show the head of the data



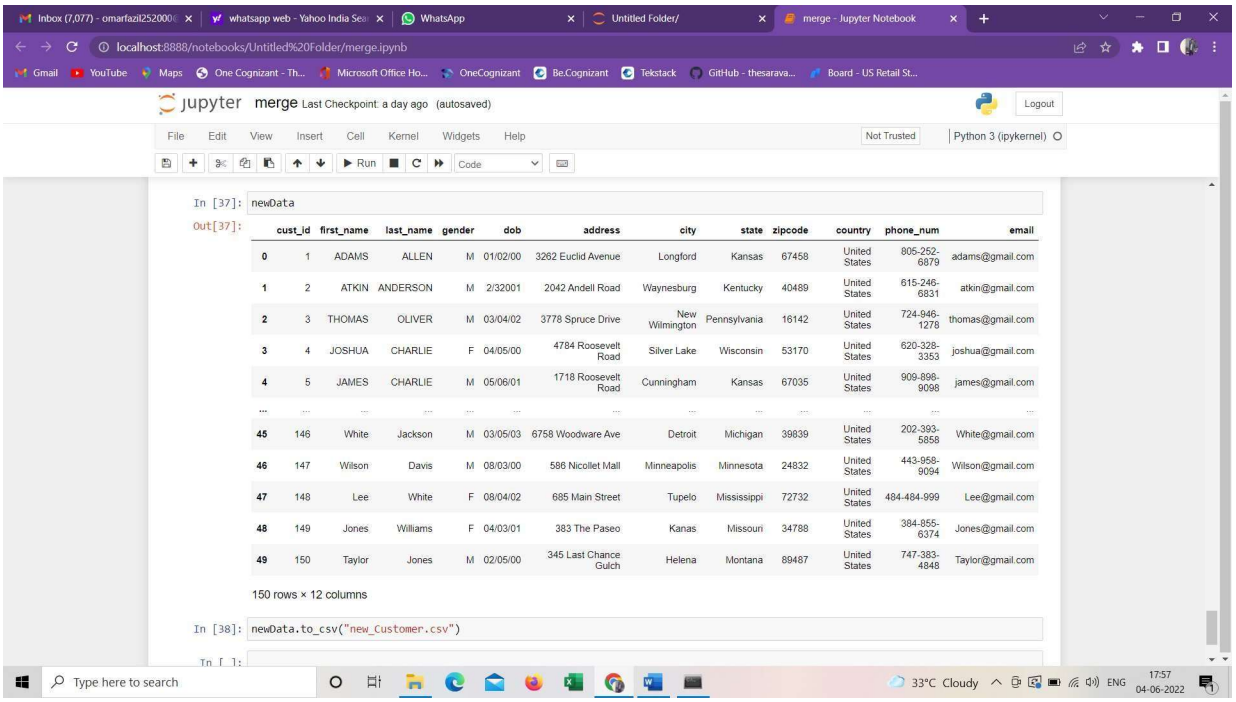
```
In [28]: df_ = df1.merge(df2,axis).merge(df3, on='cust_id')

In [31]: frame = [df1,df2,df3]
         newData = pd.concat(frame)

In [32]: newData.head()
```

Out[32]:

	cust_id	first_name	last_name	gender	dob	address	city	state	zipcode	country	phone_num	email
0	1	ADAMS	ALLEN	M	01/02/00	3262 Euclid Avenue	Longford	Kansas	67458	United States	805-252-6879	adams@gmail.com
1	2	ATKIN	ANDERSON	M	2/32001	2042 Andell Road	Waynesburg	Kentucky	40489	United States	615-246-6831	atkin@gmail.com
2	3	THOMAS	OLIVER	M	03/04/02	3778 Spruce Drive	New Wilmington	Pennsylvania	16142	United States	724-946-1278	thomas@gmail.com
3	4	JOSHUA	CHARLIE	F	04/05/00	4784 Roosevelt Road	Silver Lake	Wisconsin	53170	United States	620-328-3353	joshua@gmail.com
4	5	JAMES	CHARLIE	M	05/06/01	1718 Roosevelt Road	Cunningham	Kansas	67035	United States	909-898-9098	james@gmail.com

```
In [33]: newData.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 49
```

Saving the cleaned and pre-processed customer



In [37]: newData

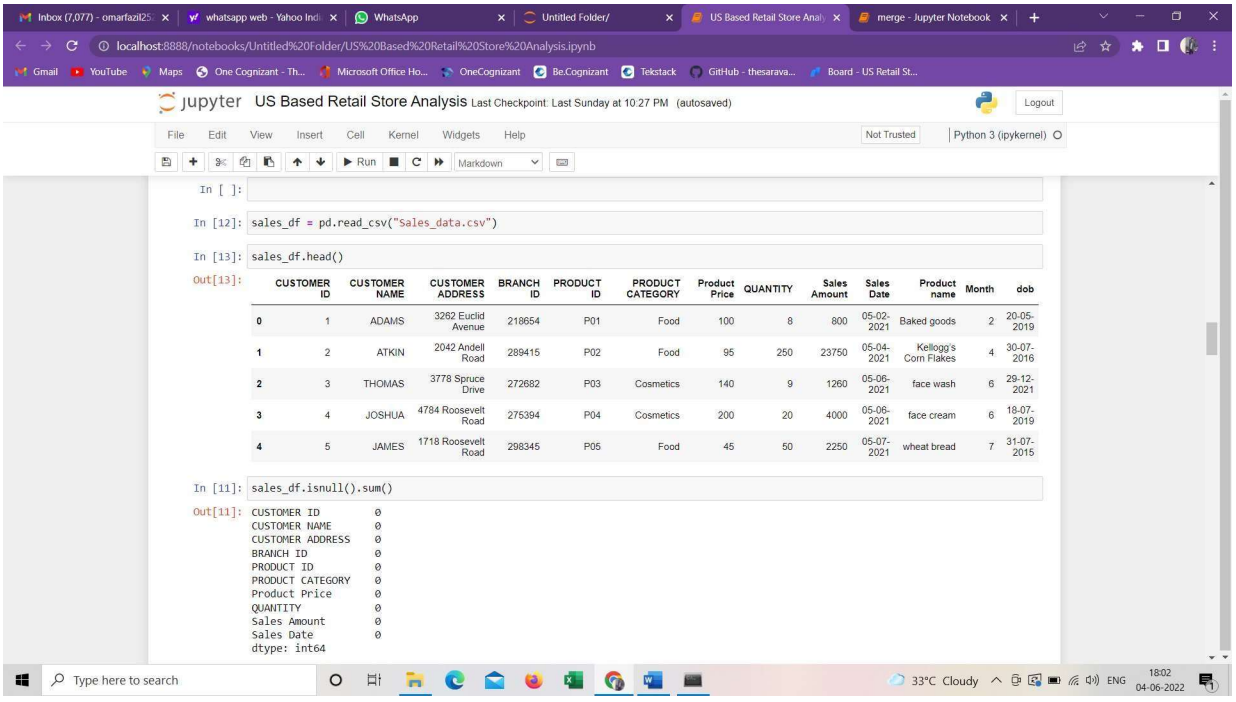
Out[37]:

	cust_id	first_name	last_name	gender	dob	address	city	state	zipcode	country	phone_num	email
0	1	ADAMS	ALLEN	M	01/02/00	3262 Euclid Avenue	Longford	Kansas	67458	United States	805-252-6879	adams@gmail.com
1	2	ATKIN	ANDERSON	M	2/32001	2042 Andell Road	Waynesburg	Kentucky	40489	United States	615-246-6831	atkin@gmail.com
2	3	THOMAS	OLIVER	M	03/04/02	3778 Spruce Drive	New Wilmington	Pennsylvania	16142	United States	724-946-1278	thomas@gmail.com
3	4	JOSHUA	CHARLIE	F	04/05/00	4784 Roosevelt Road	Silver Lake	Wisconsin	53170	United States	620-328-3353	joshua@gmail.com
4	5	JAMES	CHARLIE	M	05/06/01	1718 Roosevelt Road	Cunningham	Kansas	67035	United States	909-898-9098	james@gmail.com
...	...	...	...	...	...	...	...	...	...	...	...	...
45	146	White	Jackson	M	03/05/03	6758 Woodware Ave	Detroit	Michigan	39839	United States	202-393-5858	White@gmail.com
46	147	Wilson	Davis	M	08/03/00	586 Nicollet Mall	Minneapolis	Minnesota	24832	United States	443-958-9094	Wilson@gmail.com
47	148	Lee	White	F	08/04/02	685 Main Street	Tupelo	Mississippi	72732	United States	484-484-999	Lee@gmail.com
48	149	Jones	Williams	F	04/03/01	383 The Paseo	Kanas	Missouri	34788	United States	384-855-6374	Jones@gmail.com
49	150	Taylor	Jones	M	02/05/00	345 Last Chance Gulch	Helena	Montana	89487	United States	747-383-4848	Taylor@gmail.com

150 rows × 12 columns

In [38]: newData.to_csv("new_Customer.csv")

4. Import sales data and check if there are any null values present in the data



In [ ]:

In [12]: sales_df = pd.read_csv("Sales_data.csv")

In [13]: sales_df.head()

Out[13]:

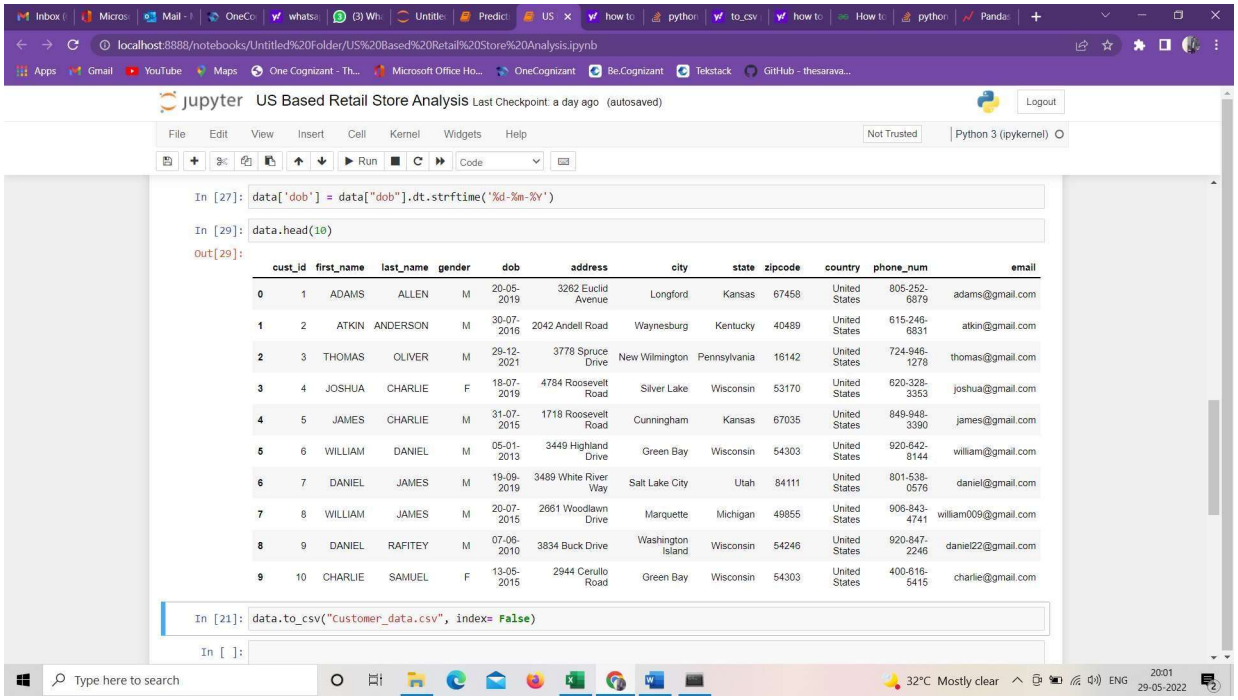
	CUSTOMER ID	CUSTOMER NAME	CUSTOMER ADDRESS	BRANCH ID	PRODUCT ID	PRODUCT CATEGORY	Product Price	QUANTITY	Sales Amount	Sales Date	Product name	Month	dob
0	1	ADAMS	3262 Euclid Avenue	218654	P01	Food	100	8	800	05-02-2021	Baked goods	2	20-05-2019
1	2	ATKIN	2042 Andell Road	289415	P02	Food	95	250	23750	05-04-2021	Kellogg's Corn Flakes	4	30-07-2016
2	3	THOMAS	3778 Spruce Drive	272682	P03	Cosmetics	140	9	1260	05-06-2021	face wash	6	29-12-2021
3	4	JOSHUA	4784 Roosevelt Road	275394	P04	Cosmetics	200	20	4000	05-06-2021	face cream	6	18-07-2019
4	5	JAMES	1718 Roosevelt Road	298345	P05	Food	45	50	2250	05-07-2021	wheat bread	7	31-07-2015

In [11]: sales_df.isnull().sum()

Out[11]:

```
CUSTOMER ID         0
CUSTOMER NAME       0
CUSTOMER ADDRESS    0
BRANCH ID           0
PRODUCT ID          0
PRODUCT CATEGORY    0
Product Price       0
QUANTITY            0
Sales Amount        0
Sales Date          0
dtype: int64
```

5. The date format had been changed from dd/mm/yyyy to mm/dd/yyyy using pandas and the snap of the formatted data is pinged below.



jupyter US Based Retail Store Analysis Last Checkpoint: a day ago (autosaved)

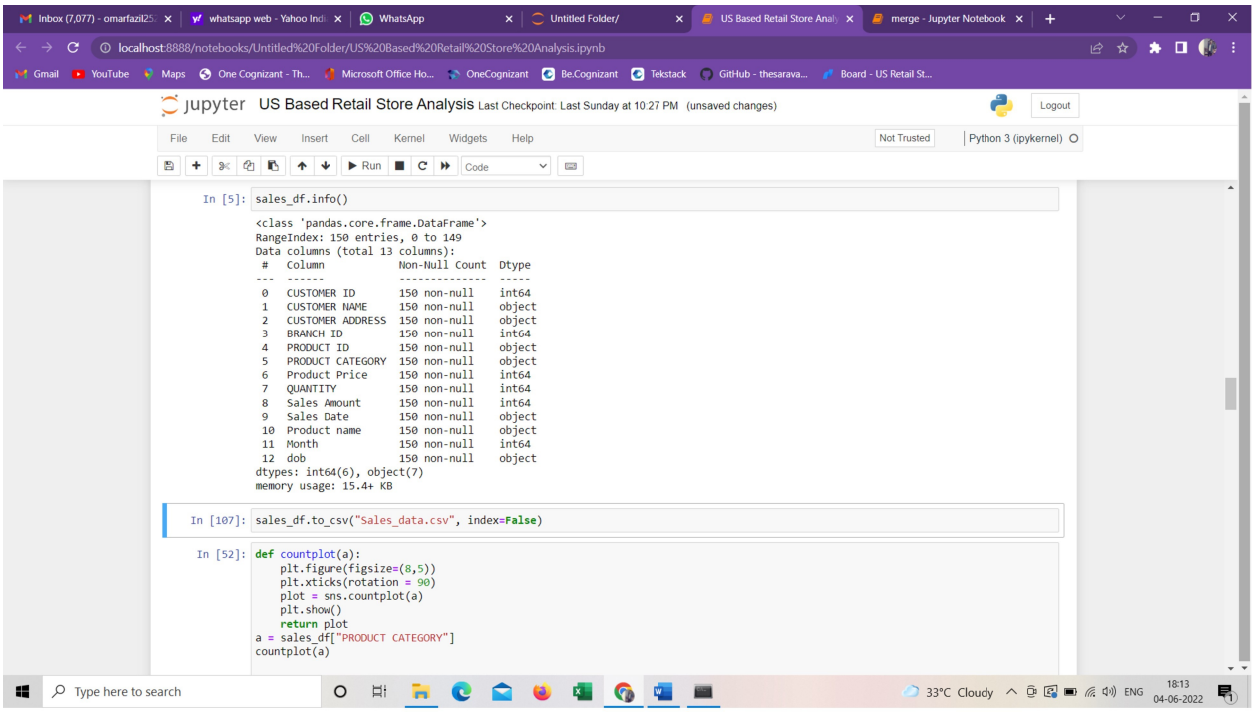
```
In [27]: data['dob'] = data["dob"].dt.strftime('%d-%m-%Y')

In [29]: data.head(10)
```

Out[29]:

	cust_id	first_name	last_name	gender	dob	address	city	state	zipcode	country	phone_num	email
0	1	ADAMS	ALLEN	M	20-05-2019	3262 Euclid Avenue	Longford	Kansas	67458	United States	805-252-6879	adams@gmail.com
1	2	ATKIN	ANDERSON	M	30-07-2016	2042 Andell Road	Waynesburg	Kentucky	40489	United States	615-246-6831	atkin@gmail.com
2	3	THOMAS	OLIVER	M	29-12-2021	3778 Spruce Drive	New Wilmington	Pennsylvania	16142	United States	724-946-1278	thomas@gmail.com
3	4	JOSHUA	CHARLIE	F	18-07-2019	4784 Roosevelt Road	Silver Lake	Wisconsin	53170	United States	620-328-3353	joshua@gmail.com
4	5	JAMES	CHARLIE	M	31-07-2015	1718 Roosevelt Road	Cunningham	Kansas	67035	United States	849-948-3390	james@gmail.com
5	6	WILLIAM	DANIEL	M	05-01-2013	3449 Highland Drive	Green Bay	Wisconsin	54303	United States	920-842-8144	william@gmail.com
6	7	DANIEL	JAMES	M	19-09-2019	3489 White River Way	Salt Lake City	Utah	84111	United States	801-538-0576	daniel@gmail.com
7	8	WILLIAM	JAMES	M	20-07-2015	2661 Woodlawn Drive	Marquette	Michigan	49855	United States	906-843-4741	william009@gmail.com
8	9	DANIEL	RAFITEY	M	07-06-2010	3834 Buck Drive	Washington Island	Wisconsin	54246	United States	920-847-2246	daniel22@gmail.com
9	10	CHARLIE	SAMUEL	F	13-05-2015	2944 Cerullo Road	Green Bay	Wisconsin	54303	United States	400-616-5415	charlie@gmail.com

```
In [21]: data.to_csv("Customer_data.csv", index= False)

In [ ]:
```

3. Saving the cleaned and pre-processed customer and sales data

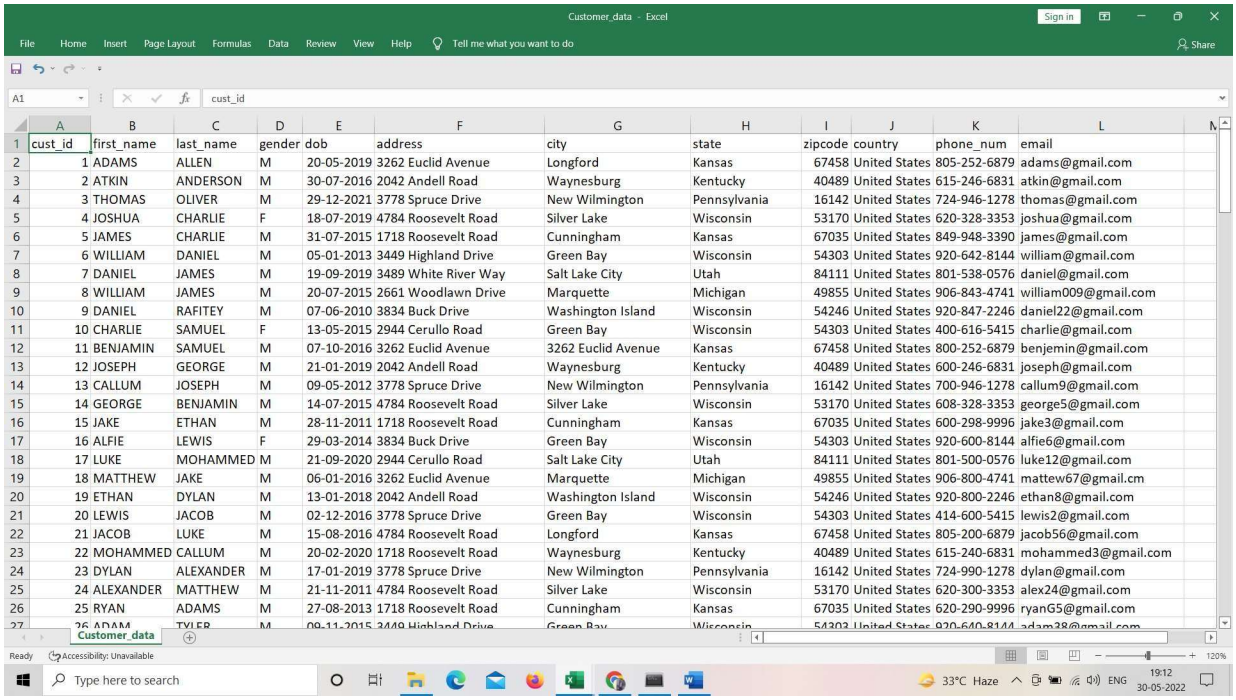


jupyter US Based Retail Store Analysis Last Checkpoint: Last Sunday at 10:27 PM (unsaved changes)

```
In [5]: sales_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   CUSTOMER ID       150 non-null    int64
 1   CUSTOMER NAME     150 non-null    object
 2   CUSTOMER ADDRESS  150 non-null    object
 3   BRANCH ID         150 non-null    int64
 4   PRODUCT ID        150 non-null    object
 5   PRODUCT CATEGORY  150 non-null    object
 6   Product Price     150 non-null    int64
 7   QUANTITY          150 non-null    int64
 8   Sales Amount      150 non-null    int64
 9   Sales Date        150 non-null    object
 10  Product name      150 non-null    object
 11  Month             150 non-null    int64
 12  dob               150 non-null    object
dtypes: int64(6), object(7)
memory usage: 15.4+ KB

In [107]: sales_df.to_csv("Sales_data.csv", index=False)

In [52]: def countplot(a):
             plt.figure(figsize=(8,5))
             plt.xticks(rotation = 90)
             plot = sns.countplot(a)
             plt.show()
             return plot
         a = sales_df["PRODUCT CATEGORY"]
         countplot(a)
```

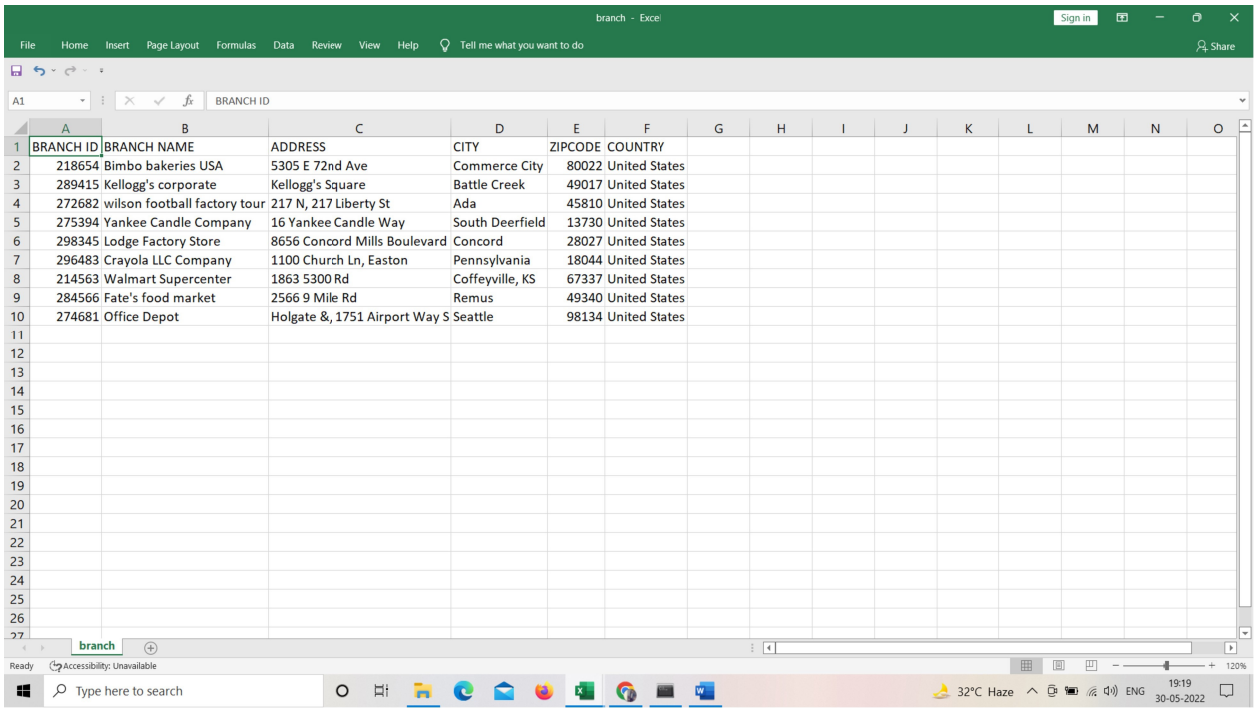
5.1.2 Source Data

Customer data



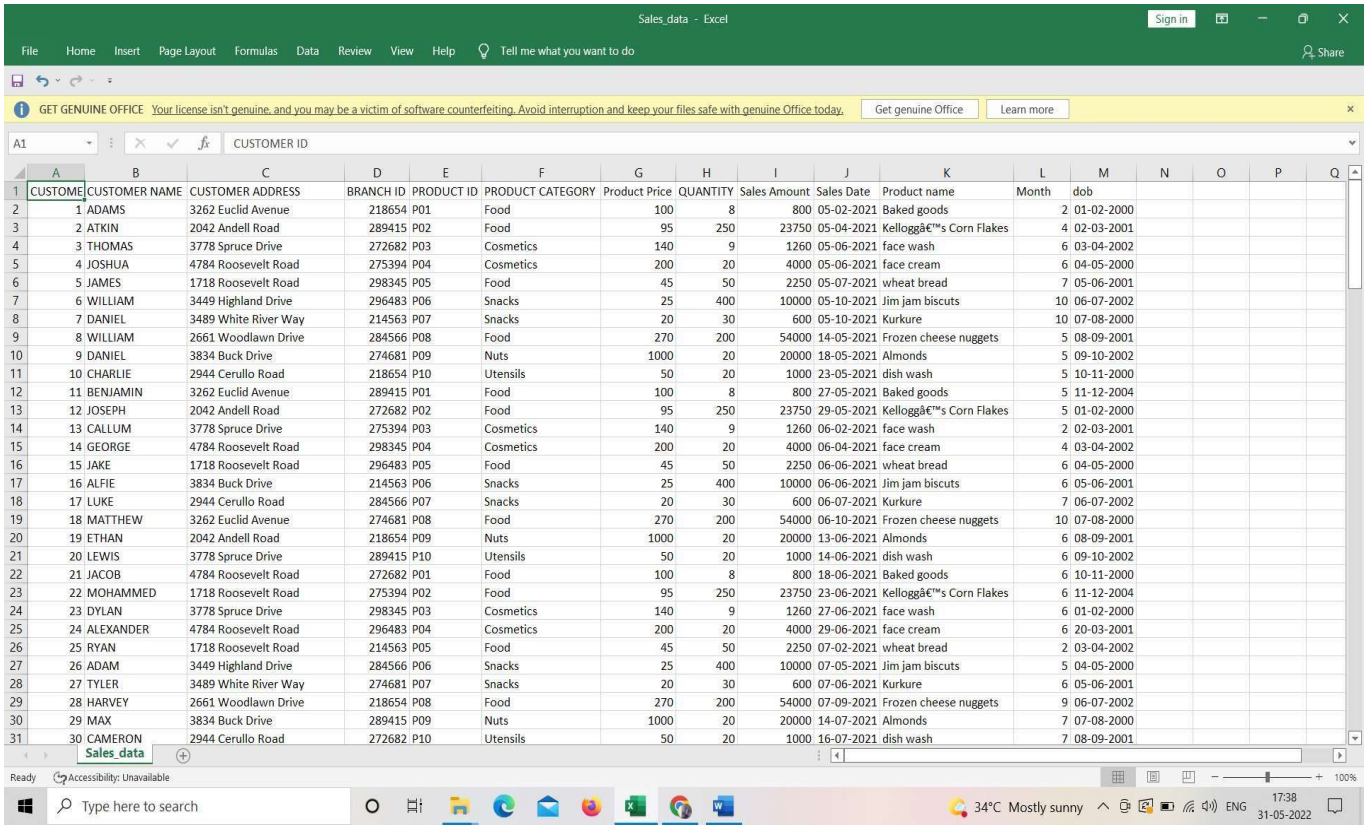
cust_id	first_name	last_name	gender	dob	address	city	state	zipcode	country	phone_num	email
1	ADAMS	ALLEN	M	20-05-2019	3262 Euclid Avenue	Longford	Kansas	67458	United States	805-252-6879	adams@gmail.com
2	ATKIN	ANDERSON	M	30-07-2016	2042 Andell Road	Waynesburg	Kentucky	40489	United States	615-246-6831	atkin@gmail.com
3	THOMAS	OLIVER	M	29-12-2021	3778 Spruce Drive	New Wilmington	Pennsylvania	16142	United States	724-946-1278	thomas@gmail.com
4	JOSHUA	CHARLIE	F	18-07-2019	4784 Roosevelt Road	Silver Lake	Wisconsin	53170	United States	620-328-3353	joshua@gmail.com
5	JAMES	CHARLIE	M	31-07-2015	1718 Roosevelt Road	Cunningham	Kansas	67035	United States	849-948-3390	james@gmail.com
6	WILLIAM	DANIEL	M	05-01-2013	3449 Highland Drive	Green Bay	Wisconsin	54303	United States	920-642-8144	william@gmail.com
7	DANIEL	JAMES	M	19-09-2019	3489 White River Way	Salt Lake City	Utah	84111	United States	801-538-0576	daniel@gmail.com
8	WILLIAM	JAMES	M	20-07-2015	2661 Woodlawn Drive	Marquette	Michigan	49855	United States	906-843-4741	william009@gmail.com
9	DANIEL	RAFITEY	M	07-06-2010	3834 Buck Drive	Washington Island	Wisconsin	54246	United States	920-847-2246	daniel22@gmail.com
10	CHARLIE	SAMUEL	F	13-05-2015	2944 Cerullo Road	Green Bay	Wisconsin	54303	United States	400-616-5415	charlie@gmail.com
11	BENJAMIN	SAMUEL	M	07-10-2016	3262 Euclid Avenue	3262 Euclid Avenue	Kansas	67458	United States	800-252-6879	benjemin@gmail.com
12	JOSEPH	GEORGE	M	21-01-2019	2042 Andell Road	Waynesburg	Kentucky	40489	United States	600-246-6831	joseph@gmail.com
13	CALLUM	JOSEPH	M	09-05-2012	3778 Spruce Drive	New Wilmington	Pennsylvania	16142	United States	700-946-1278	callum9@gmail.com
14	GEORGE	BENJAMIN	M	14-07-2015	4784 Roosevelt Road	Silver Lake	Wisconsin	53170	United States	608-328-3353	george5@gmail.com
15	JAKE	ETHAN	M	28-11-2011	1718 Roosevelt Road	Cunningham	Kansas	67035	United States	600-298-9996	jake3@gmail.com
16	ALFIE	LEWIS	F	29-03-2014	3834 Buck Drive	Green Bay	Wisconsin	54303	United States	920-600-8144	alfie6@gmail.com
17	LUKE	MOHAMMED	M	21-09-2020	2944 Cerullo Road	Salt Lake City	Utah	84111	United States	801-500-0576	luke12@gmail.com
18	MATTHEW	JAKE	M	06-01-2016	3262 Euclid Avenue	Marquette	Michigan	49855	United States	906-800-4741	mattew67@gmail.cm
19	ETHAN	DYLAN	M	13-01-2018	2042 Andell Road	Washington Island	Wisconsin	54246	United States	920-800-2246	ethan8@gmail.com
20	LEWIS	JACOB	M	02-12-2016	3778 Spruce Drive	Green Bay	Wisconsin	54303	United States	414-600-5415	lewis2@gmail.com
21	JACOB	LUKE	M	15-08-2016	4784 Roosevelt Road	Longford	Kansas	67458	United States	805-200-6879	jacob56@gmail.com
22	MOHAMMED	CALLUM	M	20-02-2020	1718 Roosevelt Road	Waynesburg	Kentucky	40489	United States	615-240-6831	mohammed3@gmail.com
23	DYLAN	ALEXANDER	M	17-01-2019	3778 Spruce Drive	New Wilmington	Pennsylvania	16142	United States	724-990-1278	dylan@gmail.com
24	ALEXANDER	MATTHEW	M	21-11-2011	4784 Roosevelt Road	Silver Lake	Wisconsin	53170	United States	620-300-3353	alex24@gmail.com
25	RYAN	ADAMS	M	27-08-2013	1718 Roosevelt Road	Cunningham	Kansas	67035	United States	620-290-9996	ryanG5@gmail.com

Branch data



BRANCH ID	BRANCH NAME	ADDRESS	CITY	ZIPCODE	COUNTRY
218654	Bimbo bakeries USA	5305 E 72nd Ave	Commerce City	80022	United States
289415	Kellogg's corporate	Kellogg's Square	Battle Creek	49017	United States
272682	wilson football factory tour	217 N, 217 Liberty St	Ada	45810	United States
275394	Yankee Candle Company	16 Yankee Candle Way	South Deerfield	13730	United States
298345	Lodge Factory Store	8656 Concord Mills Boulevard	Concord	28027	United States
296483	Crayola LLC Company	1100 Church Ln, Easton	Pennsylvania	18044	United States
214563	Walmart Supercenter	1863 5300 Rd	Coffeyville, KS	67337	United States
284566	Fate's food market	2566 9 Mile Rd	Remus	49340	United States
274681	Office Depot	Holgate &, 1751 Airport Way S	Seattle	98134	United States

Sales data



CUSTOME	CUSTOMER NAME	CUSTOMER ADDRESS	BRANCH ID	PRODUCT ID	PRODUCT CATEGORY	Product Price	QUANTITY	Sales Amount	Sales Date	Product name	Month	dob
1	ADAMS	3262 Euclid Avenue	218654	P01	Food	100	8	800	05-02-2021	Baked goods	2	01-02-2000
2	ATKIN	2042 Andell Road	289415	P02	Food	95	250	23750	05-04-2021	Kelloggâ€™s Corn Flakes	4	02-03-2001
3	THOMAS	3778 Spruce Drive	272682	P03	Cosmetics	140	9	1260	05-06-2021	face wash	6	03-04-2002
4	JOSHUA	4784 Roosevelt Road	275394	P04	Cosmetics	200	20	4000	05-06-2021	face cream	6	04-05-2000
5	JAMES	1718 Roosevelt Road	298345	P05	Food	45	50	2250	05-07-2021	wheat bread	7	05-06-2001
6	WILLIAM	3449 Highland Drive	296483	P06	Snacks	25	400	10000	05-10-2021	Jim jam biscuits	10	06-07-2002
7	DANIEL	3489 White River Way	214563	P07	Snacks	20	30	600	05-10-2021	Kurkure	10	07-08-2000
8	WILLIAM	2661 Woodlawn Drive	284566	P08	Food	270	200	54000	14-05-2021	Frozen cheese nuggets	5	08-09-2001
9	DANIEL	3834 Buck Drive	274681	P09	Nuts	1000	20	20000	18-05-2021	Almonds	5	09-10-2002
10	CHARLIE	2944 Cerullo Road	218654	P10	Utensils	50	20	1000	23-05-2021	dish wash	5	10-11-2000
11	BENJAMIN	3262 Euclid Avenue	289415	P01	Food	100	8	800	27-05-2021	Baked goods	5	11-12-2004
12	JOSEPH	2042 Andell Road	272682	P02	Food	95	250	23750	29-05-2021	Kelloggâ€™s Corn Flakes	5	01-02-2000
13	CALLUM	3778 Spruce Drive	275394	P03	Cosmetics	140	9	1260	06-02-2021	face wash	2	02-03-2001
14	GEORGE	4784 Roosevelt Road	298345	P04	Cosmetics	200	20	4000	06-04-2021	face cream	4	03-04-2002
15	JAKE	1718 Roosevelt Road	296483	P05	Food	45	50	2250	06-06-2021	wheat bread	6	04-05-2000
16	ALFIE	3834 Buck Drive	214563	P06	Snacks	25	400	10000	06-06-2021	Jim jam biscuits	6	05-06-2001
17	LUKE	2944 Cerullo Road	284566	P07	Snacks	20	30	600	06-07-2021	Kurkure	7	06-07-2002
18	MATTHEW	3262 Euclid Avenue	274681	P08	Food	270	200	54000	06-10-2021	Frozen cheese nuggets	10	07-08-2000
19	ETHAN	2042 Andell Road	218654	P09	Nuts	1000	20	20000	13-06-2021	Almonds	6	08-09-2001
20	LEWIS	3778 Spruce Drive	289415	P10	Utensils	50	20	1000	14-06-2021	dish wash	6	09-10-2002
21	JACOB	4784 Roosevelt Road	272682	P01	Food	100	8	800	18-06-2021	Baked goods	6	10-11-2000
22	MOHAMMED	1718 Roosevelt Road	275394	P02	Food	95	250	23750	23-06-2021	Kelloggâ€™s Corn Flakes	6	11-12-2004
23	DYLAN	3778 Spruce Drive	298345	P03	Cosmetics	140	9	1260	27-06-2021	face wash	6	01-02-2000
24	ALEXANDER	4784 Roosevelt Road	296483	P04	Cosmetics	200	20	4000	29-06-2021	face cream	6	20-03-2001
25	RYAN	1718 Roosevelt Road	214563	P05	Food	45	50	2250	07-02-2021	wheat bread	2	03-04-2002
26	ADAM	3449 Highland Drive	284566	P06	Snacks	25	400	10000	07-05-2021	Jim jam biscuits	5	04-05-2000
27	TYLER	3489 White River Way	274681	P07	Snacks	20	30	600	07-06-2021	Kurkure	6	05-06-2001
28	HARVEY	2661 Woodlawn Drive	218654	P08	Food	270	200	54000	07-09-2021	Frozen cheese nuggets	9	06-07-2002
29	MAX	3834 Buck Drive	289415	P09	Nuts	1000	20	20000	14-07-2021	Almonds	7	07-08-2000
30	CAMERON	2944 Cerullo Road	272682	P10	Utensils	50	20	1000	16-07-2021	dish wash	7	08-09-2001

5.2 MYSQL

- To handle the large subset of functionality of the huge datasets we use MySQL database for accessing the source data and transform them into a structured and cleaned data.
- MySQL works on many operating systems and with many languages including Perl, C, C++, Java, Python etc.
- MySQL works very quickly and works well even with large datasets.

1. SQL Queries

1. Importing the source data into the MySQL server

```
mysql> LOAD DATA LOCAL INFILE '/home/ubh01/Desktop/SourceData1/new_Customer.csv'
INTO TABLE customer FIELDS TERMINATED BY ',' ENCLOSED BY "" LINES TERMINATED BY
"\n" IGNORE 1 ROWS;
Query OK, 150 rows affected, 334 warnings (0.05 sec)
Records: 150  Deleted: 0  Skipped: 0  Warnings: 334
```

2. Creating Customer table in MySQL using the following command

```
mysql> create table customer(
    -> customer_id int primary key,
    -> first_name varchar(20),
    -> last_name varchar(25),
    -> gender ENUM('M','F'),
    -> dob date,
    -> address varchar(40),
    -> city varchar(30),
    -> zipcode varchar(6),
    -> country varchar(30),
    -> phone_number varchar(15),
    -> email varchar(30));
Query OK, 0 rows affected (0.15 sec)
```

3. Show the UsRetailStore database and the Customer table present in the database

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| metastore          |
| metastore_db       |
| mysql              |
| performance_schema |
| sumit              |
| sys                |
| usRetailSales      |
+--------------------+
8 rows in set (0.05 sec)

mysql> use usRetailSales;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------------+
| Tables_in_usRetailSales |
+-------------------------+
| customer                |
+-------------------------+
```

4. Show the data present in the customer table

```
mysql> select * from customer;
+-------------+------------+-----------+--------+------------+---------------------------+-----------------------+---------+-----
------+--------------+
| customer_id | first_name | last_name | gender | dob        | address                   | city                  | zipcode | coun
umber  | email        |
+-------------+------------+-----------+--------+------------+---------------------------+-----------------------+---------+-----
------+--------------+
|           1 | ADAMS      | ALLEN     | M      | 0000-00-00 | 3262 Euclid Avenue        | Longford              | Kansas  | 6745
States | 805-252-6879 |
|           2 | ATKIN      | ANDERSON  | M      | 0000-00-00 | 2042 Andell Road          | Waynesburg            | Kentuc  | 4048
States | 615-246-6831 |
|           3 | THOMAS     | OLIVER    | M      | 2003-04-02 | 3778 Spruce Drive         | New Wilmington        | Pennsy  | 1614
States | 724-946-1278 |
|           4 | JOSHUA     | CHARLIE   | F      | 0000-00-00 | 4784 Roosevelt Road       | Silver Lake           | Wiscon  | 5317
States | 620-328-3353 |
|           5 | JAMES      | CHARLIE   | M      | 2005-06-01 | 1718 Roosevelt Road       | Cunningham            | Kansas  | 6703
States | 909-898-9098 |
|           6 | WILLIAM    | DANIEL    | M      | 2006-07-02 | 3449 Highland Drive       | Green Bay             | Wiscon  | 5430
States | 920-642-8144 |
```

5. Importing the Product table and visualize the data inside the product table

```
mysql> create table product( product_id varchar(5) primary key, product_name var
char(30), product_category varchar(35), product_price int);
Query OK, 0 rows affected (0.17 sec)

mysql> LOAD DATA LOCAL INFILE '/home/ubh01/Desktop/SourceData1/product.txt' INTO
 TABLE product;
Query OK, 80 rows affected, 2838 warnings (0.05 sec)
Records: 999  Deleted: 0  Skipped: 919  Warnings: 2838

mysql> select * from product;
+------------+-------------------------------+------------------+---------------+
| product_id | product_name                  | product_category | product_price |
+------------+-------------------------------+------------------+---------------+
|            |                               |                  |             0 |
| P01        | parle biscuit                 | snack            |           100 |
| P02        | kellogs rice krispies         | snack            |           250 |
| P03        | oreo biscuit                  | snack            |           200 |
| P04        | dosritos                      | snack            |           180 |
| P05        | wellness cream fudge biscuit  | snack            |           140 |
| P06        | fruit guzher                  | snack            |           280 |
| P07        | cheetos                       | snack            |           300 |
| P08        | takis                         | snack            |           250 |
```

6. Creating branch table in MySQL and defining their respective columns and their data types

```
mysql> create table branch(
    -> branch_id int primary key,
    -> branch_name varchar(30),
    -> address varchar(180),
    -> city varchar(30),
    -> zipcode bigint,
    -> country varchar(30),
    -> CHECK (country="United States")
    -> );
Query OK, 0 rows affected (0.03 sec)
```

7. Describing the branch table

```
mysql> desc branch;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| branch_id   | int(11)      | NO   | PRI | NULL    |       |
| branch_name | varchar(30)  | YES  |     | NULL    |       |
| address     | varchar(180) | YES  |     | NULL    |       |
| city        | varchar(30)  | YES  |     | NULL    |       |
| zipcode     | bigint(20)   | YES  |     | NULL    |       |
| country     | varchar(30)  | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

5.3 Hadoop HDFS Commands

1. Initiating HDFS and YARN

```
ubh01@ubh01:~$ $HADOOP_HOME/sbin/start-dfs.sh
22/05/31 12:53:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes w
e
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/ubh01/hadoop-2.7.1/logs/hadoop-ubh01-namenode-ubh01.out
localhost: starting datanode, logging to /home/ubh01/hadoop-2.7.1/logs/hadoop-ubh01-datanode-ubh01.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/ubh01/hadoop-2.7.1/logs/hadoop-ubh01-secondarynamenode-ubh01.out
22/05/31 12:54:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes w
e
ubh01@ubh01:~$ $HADOOP_HOME/sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/ubh01/hadoop-2.7.1/logs/yarn-ubh01-resourcemanager-ubh01.out
localhost: starting nodemanager, logging to /home/ubh01/hadoop-2.7.1/logs/yarn-ubh01-nodemanager-ubh01.out
ubh01@ubh01:~$ sudo jps
[sudo] password for ubh01:
Sorry, try again.
[sudo] password for ubh01:
4209 Jps
3745 ResourceManager
3892 NodeManager
3366 DataNode
3190 NameNode
3579 SecondaryNameNode
```

2. Using SQOOP commands transferring the database to hdfs

```
ubh01@ubh01:~$ sqoop list-databases --connect jdbc:mysql://ubh01/ --username sqoop --password password
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
22/05/31 15:08:57 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
22/05/31 15:08:57 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
22/05/31 15:08:57 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is
red via the SPI and manual loading of the driver class is generally unnecessary.
information_schema
metastore
metastore_db
mysql
performance_schema
sumit
sys
usRetailSales
```

3. Using Sqoop transferring tables to HDFS

```
ubh01@ubh01:~$ sqoop list-tables --connect jdbc:mysql://ubh01/usRetailSales --username sqoop --password password
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
22/05/31 15:10:30 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
22/05/31 15:10:30 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
22/05/31 15:10:30 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is autor
red via the SPI and manual loading of the driver class is generally unnecessary.
branch
customer
product
product_review
sales
```

4. View the tables

```
mysql> show tables;
+-----------------------------+
| Tables_in_usRetailSales     |
+-----------------------------+
| branch                      |
| customer                    |
| product                     |
| product_review              |
| sales                       |
+-----------------------------+
5 rows in set (0.01 sec)
```

Using Sqoop command lets import the customer table

```
ubh01@ubh01:~$ sqoop import --connect jdbc:mysql://ubh01/usRetailSales --table c
ustomer --fields-terminated-by ',' --target-dir /user/ubh01/data_table --usernam
e sqoop --password password
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! H
Catalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! A
ccumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../zookeeper does not exist!
Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
22/05/31 15:33:00 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
22/05/31 15:33:00 WARN tool.BaseSqoopTool: Setting your password on the command-
line is insecure. Consider using -P instead.
22/05/31 15:33:01 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
22/05/31 15:33:01 INFO tool.CodeGenTool: Beginning code generation
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class
is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SP
I and manual loading of the driver class is generally unnecessary.
22/05/31 15:33:01 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
```

```
22/05/31 15:33:31 INFO mapreduce.Job:  map 0% reduce 0%
22/05/31 15:33:42 INFO mapreduce.Job:  map 25% reduce 0%
22/05/31 15:33:45 INFO mapreduce.Job:  map 50% reduce 0%
22/05/31 15:33:46 INFO mapreduce.Job:  map 75% reduce 0%
22/05/31 15:33:47 INFO mapreduce.Job:  map 100% reduce 0%
22/05/31 15:33:47 INFO mapreduce.Job: Job job_1653989566031_0002 completed succe
ssfully
22/05/31 15:33:47 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
```

5.4 Hive and Sqoop

1. Import the cust_transformation table in hdfs using sqoop command

```
(base) ubh01@ubh01:~$ sqoop import --connect jdbc:mysql://ubh01/project -table cust_transformation --fields-terminated-by ',' --us
ername sqoop --password password
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
22/06/03 22:05:56 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
22/06/03 22:05:57 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
22/06/03 22:05:57 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
22/06/03 22:05:57 INFO tool.CodeGenTool: Beginning code generation
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is autom
atically registered via the SPI and manual loading of the driver class is generally unnecessary.
22/06/03 22:05:58 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `cust_transformation` AS t LIMIT 1
22/06/03 22:05:58 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `cust_transformation` AS t LIMIT 1
22/06/03 22:05:58 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /home/ubh01/hadoop-2.7.1
Note: /tmp/sqoop-ubh01/compile/591db79b4b3d86ebb8274dd66b5e7399/cust_transformation.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
22/06/03 22:06:00 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-ubh01/compile/591db79b4b3d86ebb8274dd66b5e7399/cust_tr
ansformation.jar
22/06/03 22:06:00 WARN manager.MySQLManager: It looks like you are importing from mysql.
22/06/03 22:06:00 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
22/06/03 22:06:00 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
```

```
22/06/03 22:06:11 INFO mapreduce.Job: Running job: job_1654274141483_0001
22/06/03 22:06:23 INFO mapreduce.Job: Job job_1654274141483_0001 running in uber mode : false
22/06/03 22:06:23 INFO mapreduce.Job:  map 0% reduce 0%
22/06/03 22:06:38 INFO mapreduce.Job:  map 50% reduce 0%
22/06/03 22:06:39 INFO mapreduce.Job:  map 75% reduce 0%
22/06/03 22:06:40 INFO mapreduce.Job:  map 100% reduce 0%
22/06/03 22:06:40 INFO mapreduce.Job: Job job_1654274141483_0001 completed successfully
22/06/03 22:06:40 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=538412
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=443
                HDFS: Number of bytes written=10206
                HDFS: Number of read operations=16
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=8
        Job Counters
                Launched map tasks=4
                Other local map tasks=4
                Total time spent by all maps in occupied slots (ms)=43117
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=43117
```

```
                Other local map tasks=4
                Total time spent by all maps in occupied slots (ms)=43117
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=43117
                Total vcore-seconds taken by all map tasks=43117
                Total megabyte-seconds taken by all map tasks=44151808
        Map-Reduce Framework
                Map input records=150
                Map output records=150
                Input split bytes=443
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=1145
                CPU time spent (ms)=10560
                Physical memory (bytes) snapshot=703361024
                Virtual memory (bytes) snapshot=7699369984
                Total committed heap usage (bytes)=380633088
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=10206
22/06/03 22:06:40 INFO mapreduce.ImportJobBase: Transferred 9.9668 KB in 38.6928 seconds (263.7699 bytes/sec)
22/06/03 22:06:40 INFO mapreduce.ImportJobBase: Retrieved 150 records.
```

2. List the files in cust_transformation directory

```
(base) ubh01@ubh01:~$ hdfs dfs -ls /user/ubh01/cust_transformation
22/06/03 22:08:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
 where applicable
Found 5 items
-rw-r--r--   1 ubh01 supergroup          0 2022-06-03 22:06 /user/ubh01/cust_transformation/_SUCCESS
-rw-r--r--   1 ubh01 supergroup       2584 2022-06-03 22:06 /user/ubh01/cust_transformation/part-m-00000
-rw-r--r--   1 ubh01 supergroup       2550 2022-06-03 22:06 /user/ubh01/cust_transformation/part-m-00001
-rw-r--r--   1 ubh01 supergroup       2541 2022-06-03 22:06 /user/ubh01/cust_transformation/part-m-00002
-rw-r--r--   1 ubh01 supergroup       2531 2022-06-03 22:06 /user/ubh01/cust_transformation/part-m-00003
```

3. Import sales_transformation table in HDFS using SQOOP

```
(base) ubh01@ubh01:~$ sqoop import --connect jdbc:mysql://ubh01/project -table sales_transformation --fields-terminated-by ',' --u
sername sqoop --password password
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /home/ubh01/sqoop-1.4.7.bin__hadoop-2.6.0/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
22/06/03 22:39:02 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
22/06/03 22:39:02 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
22/06/03 22:39:02 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
22/06/03 22:39:02 INFO tool.CodeGenTool: Beginning code generation
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is autom
atically registered via the SPI and manual loading of the driver class is generally unnecessary.
22/06/03 22:39:03 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `sales_transformation` AS t LIMIT 1
22/06/03 22:39:03 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `sales_transformation` AS t LIMIT 1
22/06/03 22:39:03 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /home/ubh01/hadoop-2.7.1
Note: /tmp/sqoop-ubh01/compile/2ed48d798cda63b1bb19739b9cbf5ab2/sales_transformation.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
22/06/03 22:39:06 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-ubh01/compile/2ed48d798cda63b1bb19739b9cbf5ab2/sales_t
ransformation.jar
22/06/03 22:39:06 WARN manager.MySQLManager: It looks like you are importing from mysql.
22/06/03 22:39:06 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
22/06/03 22:39:06 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
22/06/03 22:39:06 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
```

```
22/06/03 22:40:21 INFO mapreduce.Job: The url to track the job: http://ubh01:8088/proxy/application_1654274141483_0006/
22/06/03 22:40:21 INFO mapreduce.Job: Running job: job_1654274141483_0006
22/06/03 22:40:31 INFO mapreduce.Job: Job job_1654274141483_0006 running in uber mode : false
22/06/03 22:40:31 INFO mapreduce.Job:  map 0% reduce 0%
22/06/03 22:40:38 INFO mapreduce.Job:  map 100% reduce 0%
22/06/03 22:40:39 INFO mapreduce.Job: Job job_1654274141483_0006 completed successfully
22/06/03 22:40:39 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=134729
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=87
                HDFS: Number of bytes written=9299
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
```

4. Show the location of the two targeted transformation table

```
(base) ubh01@ubh01:~$ hdfs dfs -ls /user/ubh01
22/06/03 22:41:22 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
 where applicable
Found 5 items
drwxr-xr-x   - ubh01 supergroup          0 2022-05-03 16:05 /user/ubh01/base_loudacre
drwxr-xr-x   - ubh01 supergroup          0 2022-06-03 22:06 /user/ubh01/cust_transformation
drwxr-xr-x   - ubh01 supergroup          0 2022-04-26 14:57 /user/ubh01/device
drwxr-xr-x   - ubh01 supergroup          0 2022-05-03 15:47 /user/ubh01/sales
drwxr-xr-x   - ubh01 supergroup          0 2022-06-03 22:40 /user/ubh01/sales_transformation
```

5. Create external table for cust_transformation

```
hive> CREATE EXTERNAL TABLE cust_transformation(cust_id int, name varchar(30), age int, contact varchar(60),address varchar(60))
    > row format delimited
    > fields terminated by ','
    > location '/user/ubh01/cust_transformation';
OK
Time taken: 1.118 seconds
hive> describe cust_transformation;
OK
cust_id                 int
name                    varchar(30)
age                     int
contact                 varchar(60)
address                 varchar(60)
Time taken: 0.499 seconds, Fetched: 5 row(s)
```

6. Show the data in the targeted cust_transformation table

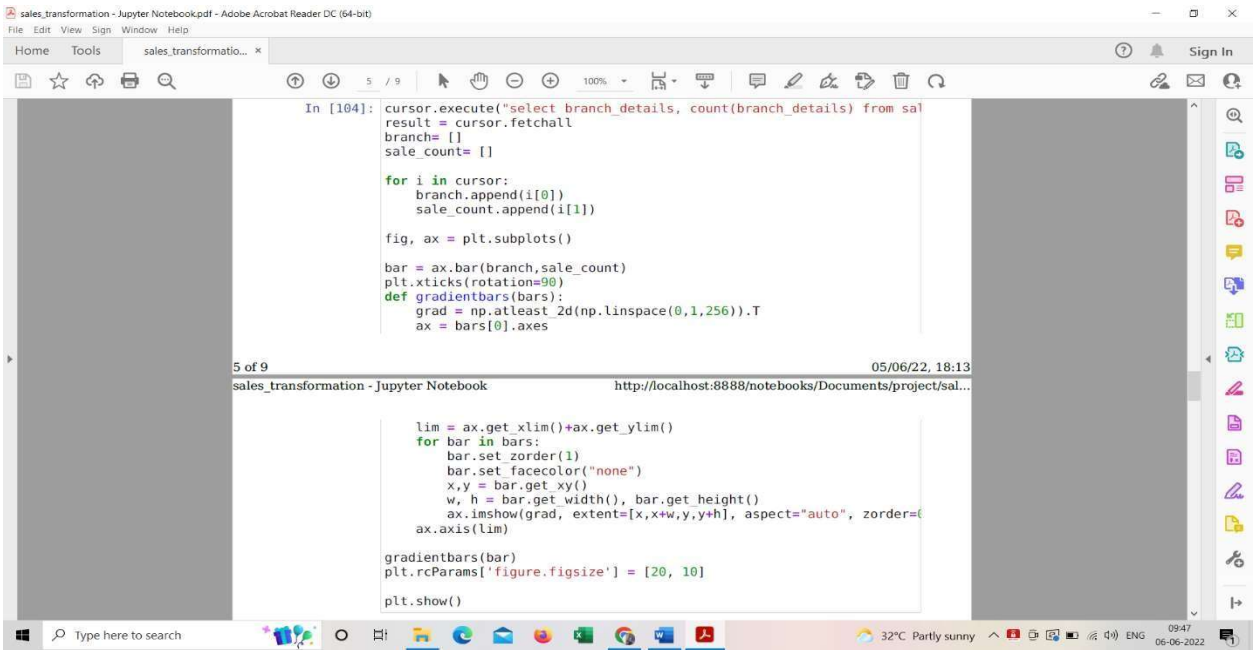
```
hive> select * from cust_transformation;
OK
1       ADAMS ALLEN     8       adams@gmail.com 805-252-6879    "3262 Euclid Avenue
2       ATKIN ANDERSON  3       atkin@gmail.com 615-246-6831    "2042 Andell Road
3       THOMAS OLIVER   24      thomas@gmail.com 724-946-1278   "3778 Spruce Drive
4       JOSHUA CHARLIE  18      joshua@gmail.com 620-328-3353   "4784 Roosevelt Road
5       JAMES CHARLIE   12      james@gmail.com NA      "1718 Roosevelt Road
6       WILLIAM DANIEL  25      william@gmail.com 920-642-8144  "3449 Highland Drive
7       DANIEL JAMES    22      daniel@gmail.com 801-538-0576   "3489 White River Way
8       WILLIAM JAMES   13      william009@gmail.com 906-843-4741       "2661 Woodlawn Drive
9       DANIEL RAFITEY  7       daniel22@gmail.com 920-847-2246 "3834 Buck Drive
```

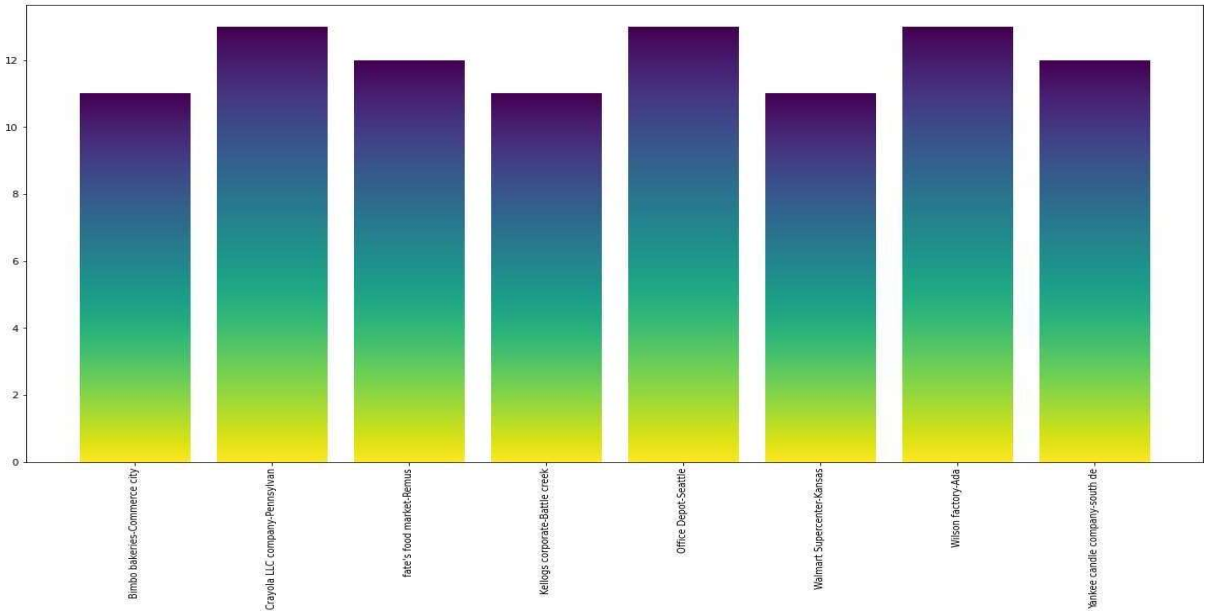
7. Create external table for targeted sales_transformation table and view the data inside the target table

```
hive> CREATE EXTERNAL TABLE sales_transformation(sale_id varchar(6),cust_id int, branch_id int, branch_details varchar(30),product
_id varchar(6),product_details varchar(30),product_price int, quantity int, sale_amount int, sales_date date, sales_year int, sale
_month varchar(30))
    > row format delimited
    > fields terminated by '/'
    > location '/user/ubh01/sales_transformation';
OK
Time taken: 0.322 seconds
hive> select * from sales_transformation;
OK
S01     1       218654  Walmart Supercenter-Kansas      P01     parle biscuit-snack     100     2       200     2022-02-15      20
22      February
S02     2       289415  Walmart Supercenter-Kansas      P02     parle biscuit-snack     250     3       750     2020-04-14      20
20      April
S03     3       272682  Walmart Supercenter-Kansas      P03     parle biscuit-snack     200     4       800     2020-11-13      20
20      November
S04     4       275394  Walmart Supercenter-Kansas      P04     parle biscuit-snack     180     2       360     2021-08-06      20
21      August
S06     6       296483  Walmart Supercenter-Kansas      P06     parle biscuit-snack     280     5       1400    2021-02-27      20
21      February
S07     7       214563  Walmart Supercenter-Kansas      P07     kellogs rice krispies-snack     300     6       1800    2021-04-18
2021    April
```

6. Output Screen

Branch Wise Sales

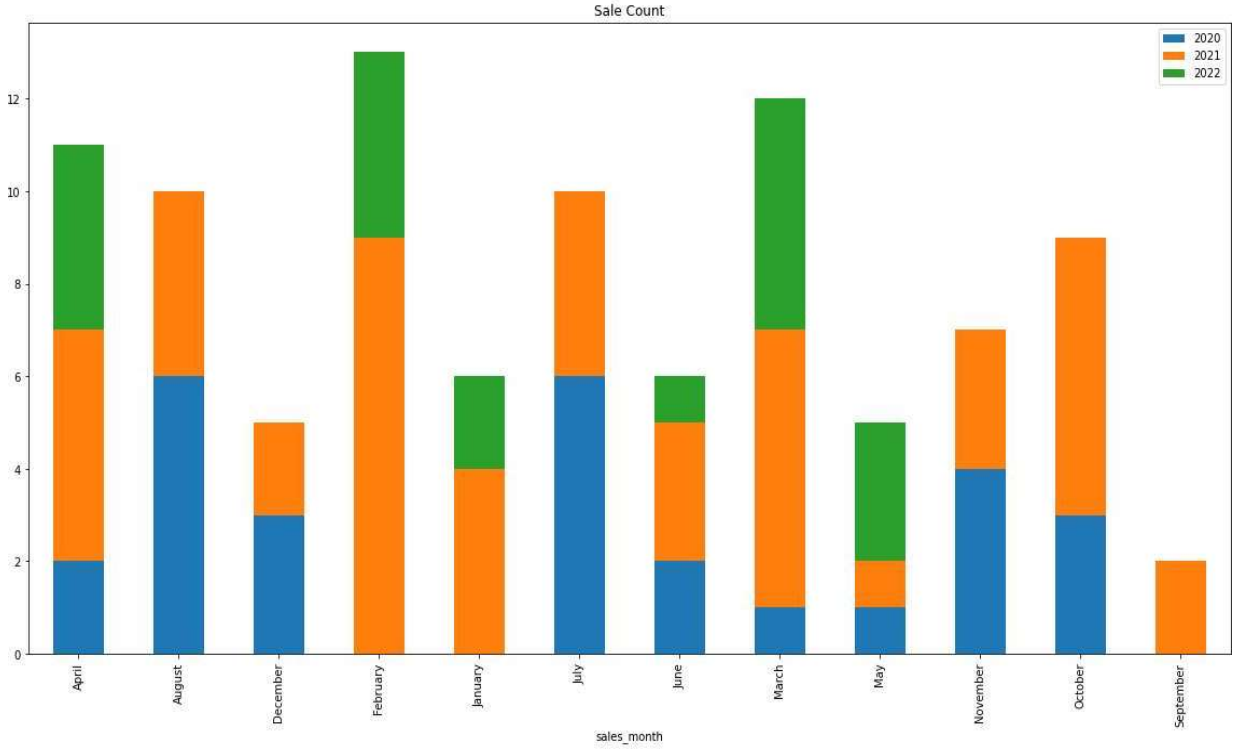




Year Wise Sales - WTR - Month

```
In [105]: query="select sales_month, sum(case when sales_year='2020' then 1 els
          df = pd.read_sql(query,conn)
          plot=df.plot.bar(title="Sale Count",x='sales_month',stacked=True);
          plt.rcParams['figure.figsize'] = [20, 20]
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```



Sales Trajectory – over the years

sales_transformation - Jupyter Notebook.pdf - Adobe Acrobat Reader DC (64-bit)

File Edit View Sign Window Help

Home Tools sales_transformatio... Sign In

8 / 9 150%

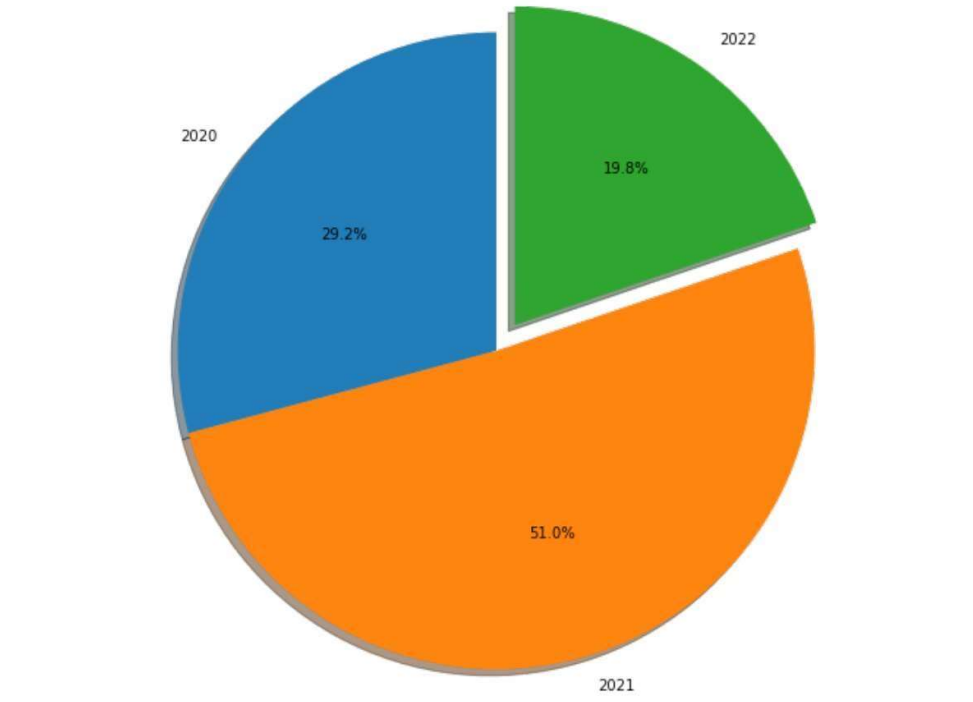
In [101]:
```
classes = '2020', '2021', '2022'
query="select count(sales_year) from sales_transformation group by sa
cursor.execute(query)
result = cursor.fetchall()
final_result = [i[0] for i in result]

explode = (0, 0, 0.1)
# only "explode" the 3rd slice (i.e. '2022')
```



```
plt.pie(final_result, explode=explode, autopct='%1.1f%%',
        labels=classes,shadow=True, startangle=90)

plt.title("Over all Sales trajectory - year")
plt.rcParams['figure.figsize'] = [20,10]
plt.show()
```

Over all Sales trajectory - year

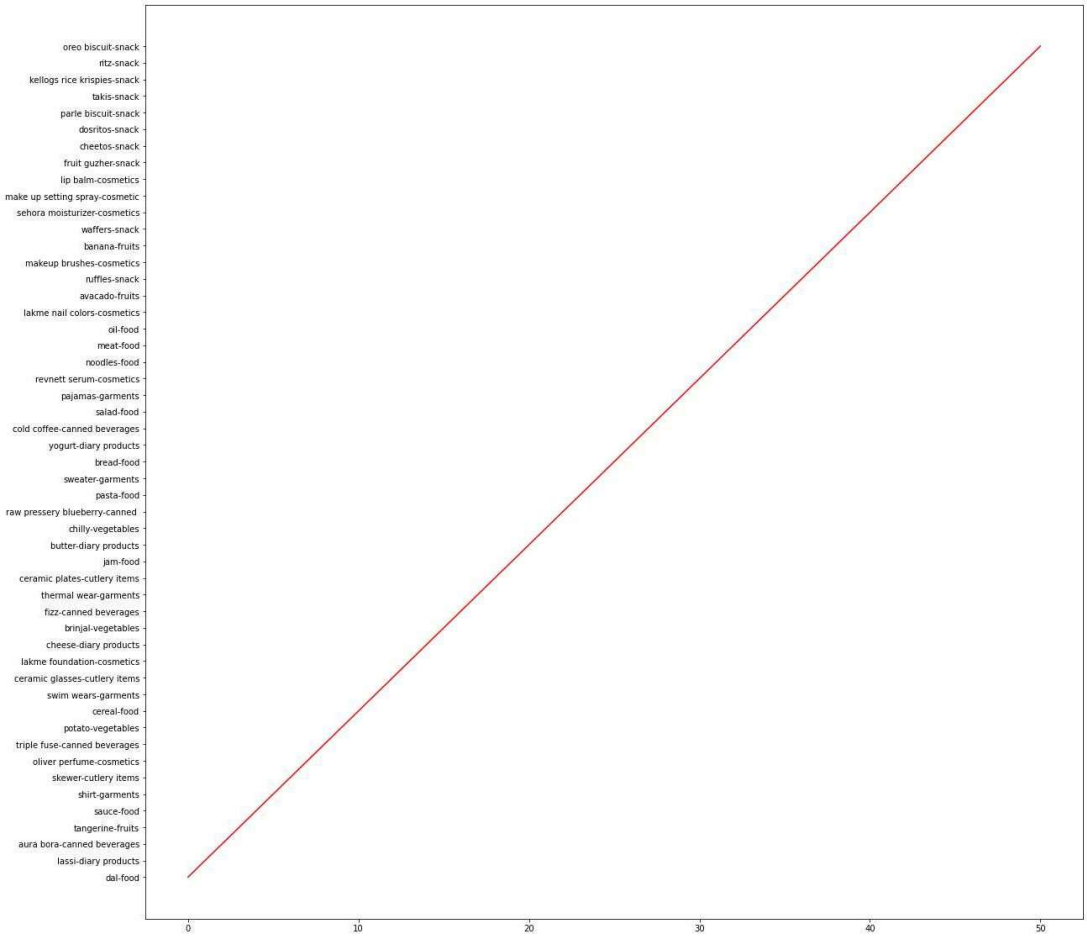
Type here to search 32°C Partly sunny ENG 09:57 06-06-2022



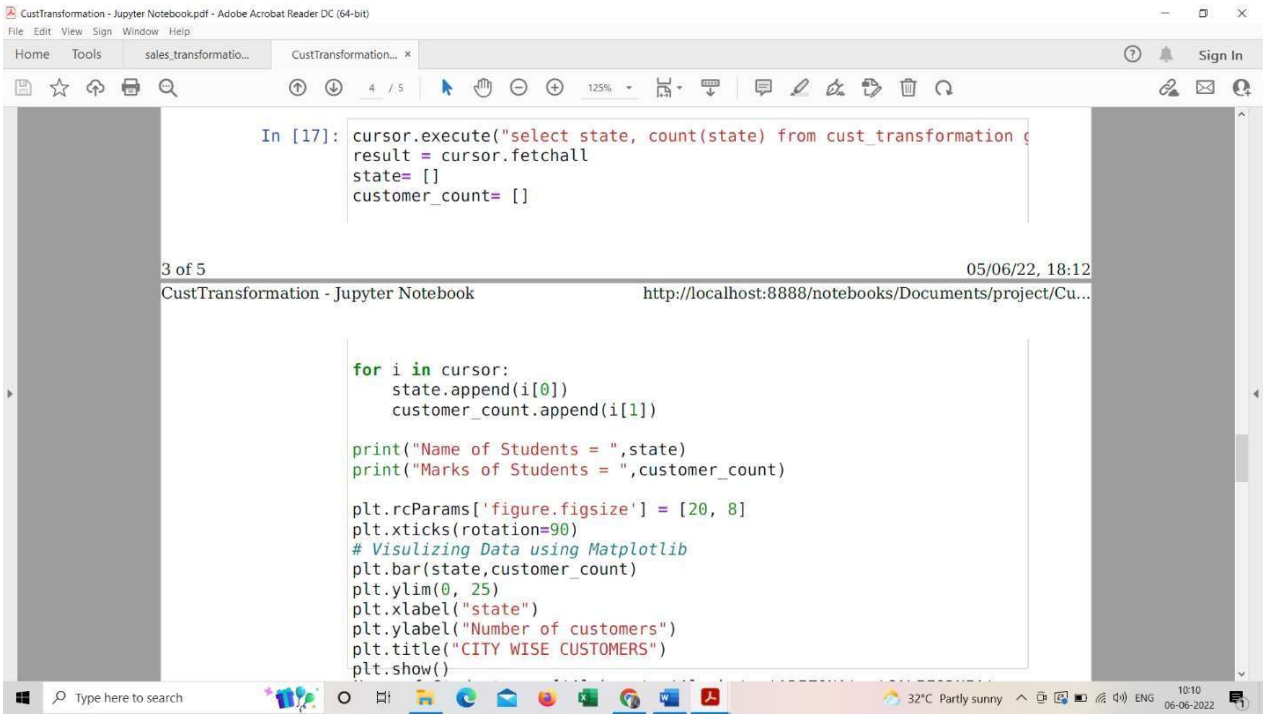
Most Sold vs Least Sold chart

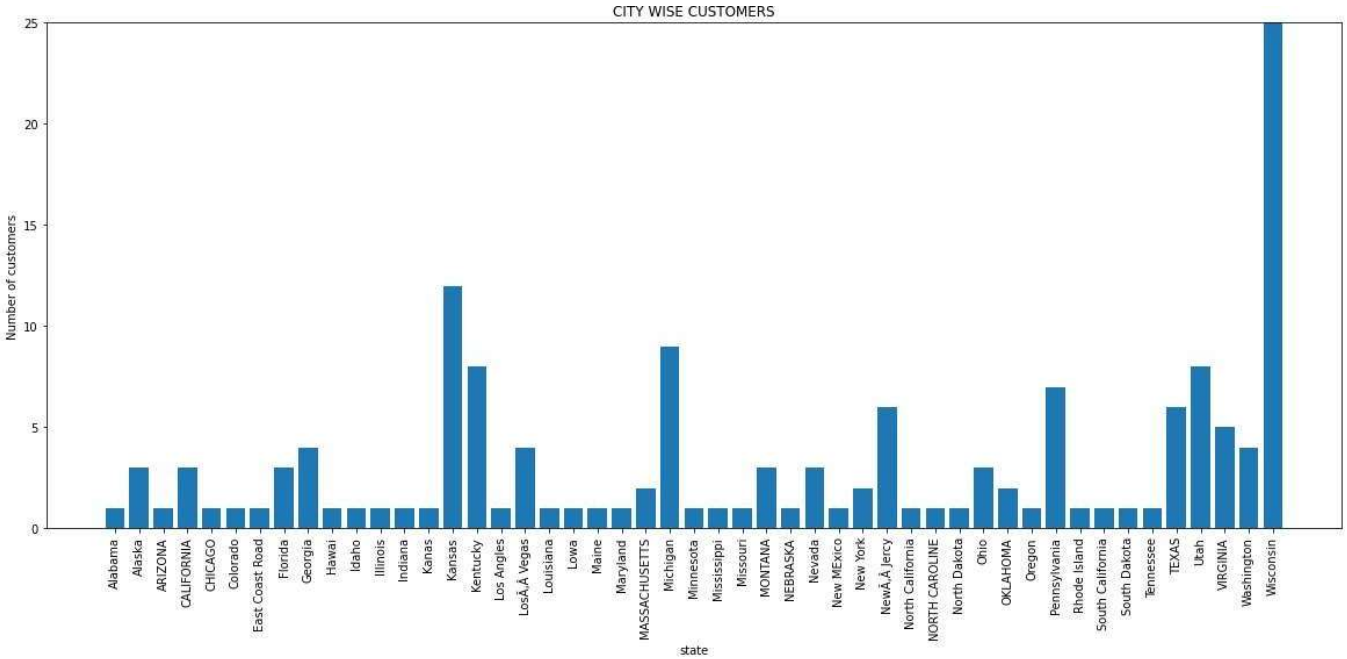
```
[107]: df=pd.read_sql_query("select product_details, count(sale_id) from sal
       plt.plot(df["product_details"],'r')
       plt.rcParams['figure.figsize'] = [20, 20]
       plt.show()
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```



Region Wise Customers





Count Of Max and Min Age of Customer – Region Wise

CustTransformation - Jupyter Notebook.pdf - Adobe Acrobat Reader DC (64-bit)

File Edit View Sign Window Help

Home Tools sales_transformatio... CustTransformation... ×

Sign In

4 / 5 200%

state

Count of Maximum and Minimum customers by age - region wise

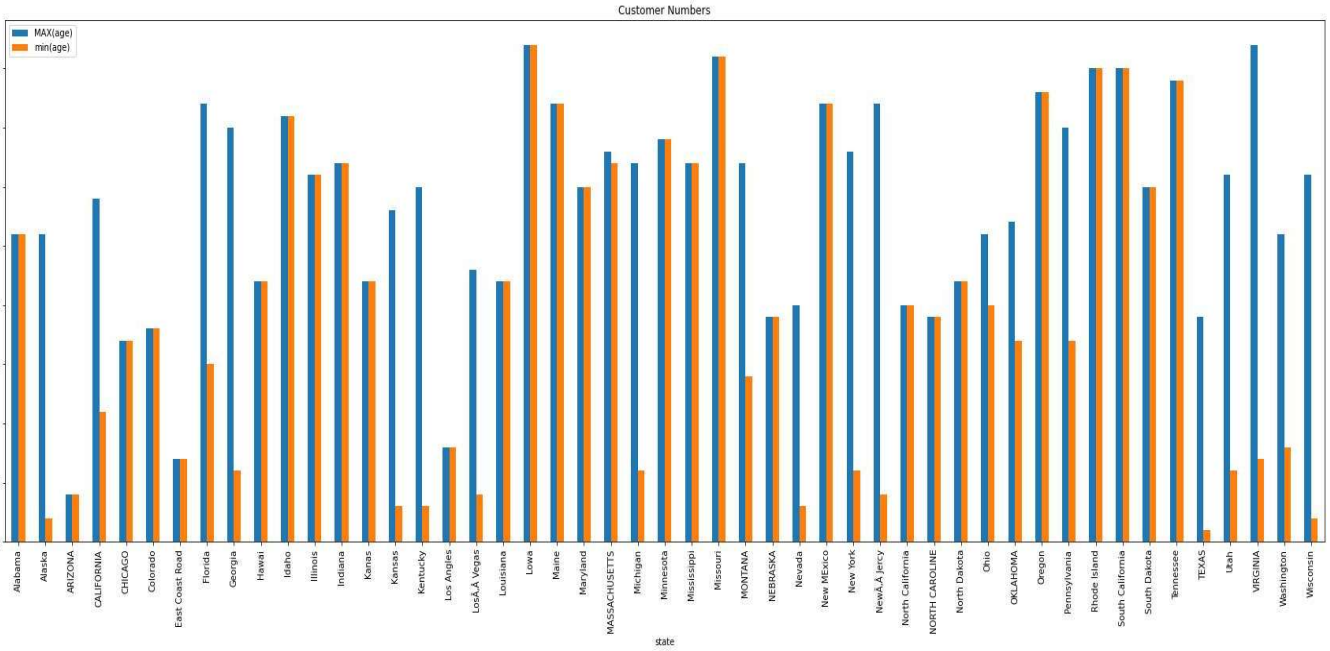
```
[18]: query="SELECT state, MAX(age), min(age) FROM cust_transformation GROU
      df = pd.read_sql(query,conn)
      plt.rcParams['figure.figsize'] = [30, 10]
      plot=df.plot.bar(title="Customer Numbers",x='state');
```

05/06/22,

ation - Jupyter Notebook http://localhost:8888/notebooks/Documents/projec

8.28 x 11.69 in

Type here to search 32°C Partly sunny ENG 10:13 06-06-2022



Customer Analysis with respective to gender

CustTransformation - Jupyter Notebook.pdf - Adobe Acrobat Reader DC (64-bit)

File Edit View Sign Window Help

Home Tools sales_transformatio... CustTransformation... Sign In

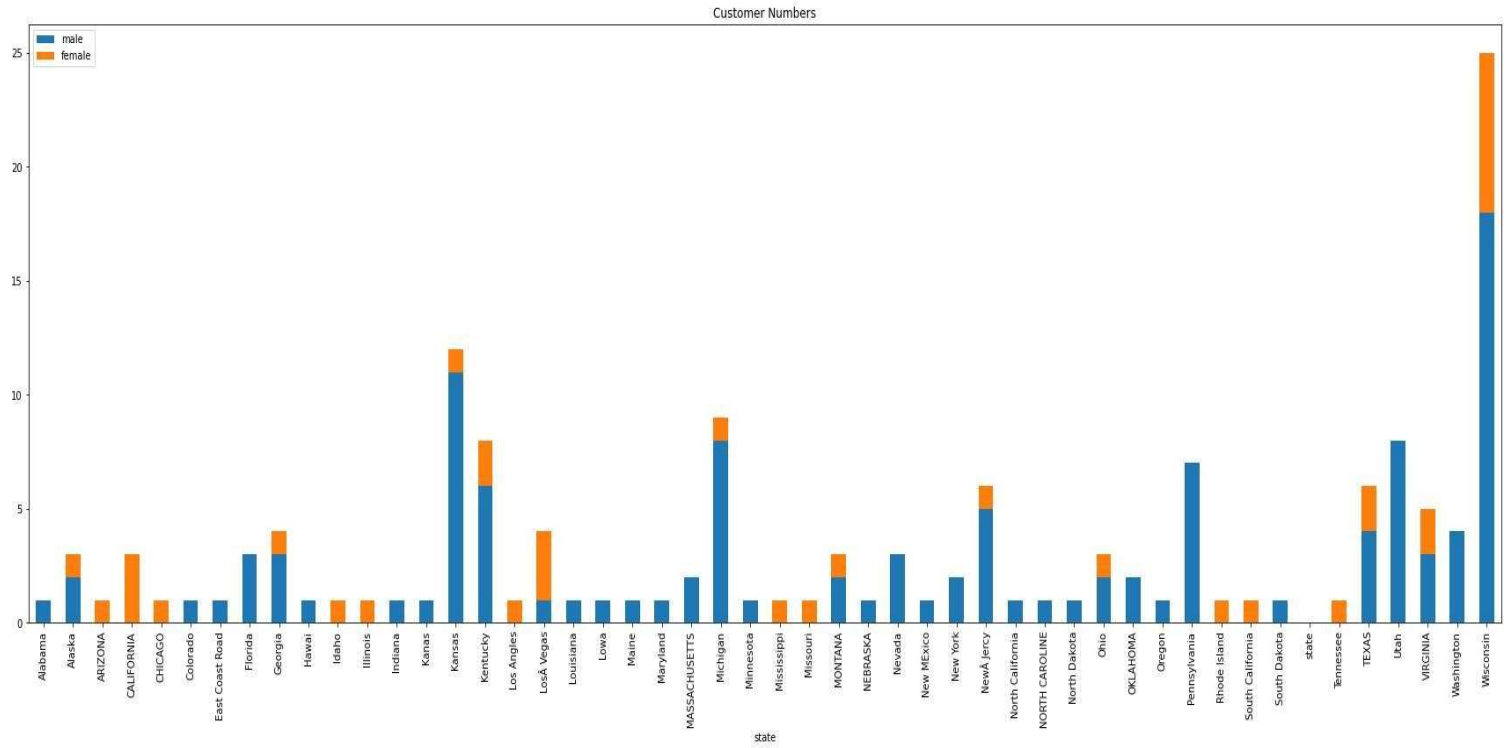
Customer Analysis - with respect to gender

In [19]:
```
query="select state,  sum(case when gender='M' then 1 else 0 end) as
df = pd.read_sql(query,conn)
plot=df.plot.bar(title="Customer Numbers",x='state',stacked=True);
fig = plot.get_figure()
fig.savefig("output3.png")
```

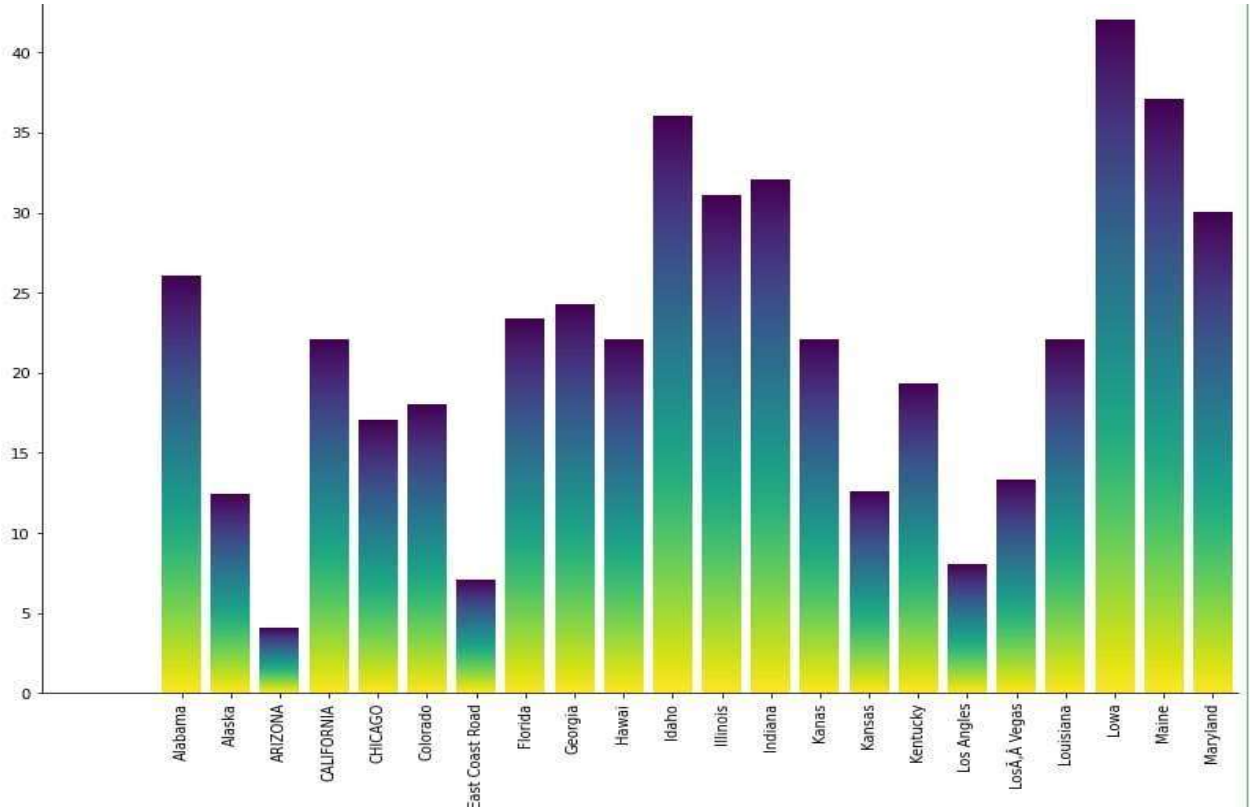
```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

8.28 x 11.69 in

Type here to search 32°C Partly sunny ENG 10:16 06-06-2022



Average Age of Customers in all Regions



CODES ASSOSIATED

In [2]:
```
import pymysql import pandas as pd import sqlalchemy
from datetime import datetime
import matplotlib.pyplot as plt
```

In [3]:
```
conn=pymysql.connect(host='localhost',port=int(3306),user='root',pas cursor=
```

In [4]:
```
df=pd.read_sql_query("SELECT * FROM customers",conn) print(df.head())
```

```
    cust_id  first_name  last_name gender         dob               a
 ddress  \
 0        0  first_name  last_name     ge  0000-00-00               a
 ddress
 1        1       ADAMS      ALLEN      M  2014-01-30   3262 Euclid
 Avenue
 2        2       ATKIN   ANDERSON      M  2019-09-12     2042 Andel
l Road
 3        3      THOMAS     OLIVER      M  1998-08-11    3778 Spruce
Drive
4        4      JOSHUA    CHARLIE      F  2004-10-02  4784 Roosevel
t Road

              city         state  zipcode        country     phone_n
um  \
0             city         state        0        country     phone_n
um
1         Longford        Kansas    67458  United States  805-252-68
79
2       Waynesburg      Kentucky    40489  United States  615-246-68
31
3  New Wilmington  Pennsylvania    16142  United States  724-946-12
78
4      Silver Lake     Wisconsin    53170  United States  620-328-33
53

                email
0             email\r
1   adams@gmail.com\r
2   atkin@gmail.com\r
3  thomas@gmail.com\r
4  joshua@gmail.com\r

/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

In [6]:
```
df["email"]=df["email"].map(lambda x: x.rstrip('\r'))
```

In [7]:
```
df["email"]
```

Out[7]:

```
0                  email
1       adams@gmail.com
2       atkin@gmail.com
3      thomas@gmail.com
4      joshua@gmail.com

White@gmail.com
147    Wilson@gmail.com
148       Lee@gmail.com
149     Jones@gmail.com
```

In [8]:
```
def age(born):
born = datetime.strptime(born, "%yyyy-%mm-%dd").date() today = date.today(
return today.year - born.year - ((today.month,
today.day) < (born.month,
born.day))
```

In [9]:
```
df["age"]=pd.read_sql_query("SELECT YEAR(CURDATE()) - YEAR(dob)from
```

146

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

In [10]:
```
df["age"].iloc[1:,].astype(int)
```

Out[10]:
```
1       8
2       3
3      24
4      18
5      12

146    31
147    34
148    32
149    41
150    31
Name: age, Length: 150, dtype: int64
```

In [11]:
```
data = pd.DataFrame({ 'cust_id':df["cust_id"],
'Name':df["first_name"]+" "+df["last_name"], 'age':df["age"],
'contact': df["email"]+" "+df["phone_num"], 'address':df['address']+"-"+df['
})
```

In [12]:
```
data=data.iloc[1:,:]
```

In [13]:
```
data
```

Out[13]:

cust_id	Name	age	contact	address	state
11	ADAMS ALLEN	8	adams@gmail.com 805-252-6879	3262 Euclid Avenue-	Kansas

cust_id	Name	age	contact	address	state
22	ATKIN ANDERSON	3	atkin@gmail.com 615-246-6831	2042 Andell Road-Waynesburg-40489	Kentucky
33	THOMAS OLIVER	24	thomas@gmail.com 724-946-1278	3778 Spruce Drive-New Wilmington-16142	Pennsylvania
44	JOSHUA CHARLIE	18	joshua@gmail.com 620-328-3353	4784 Roosevelt Road-Silver Lake-53170	Wisconsin
55	JAMES CHARLIE	12	james@gmail.com NA	1718 Roosevelt Road-Cunningham-67035	Kansas
.....	...	...	...	...	...
146146	White Jackson	31	White@gmail.com 202-393-5858	6758 Woodware Ave-Detroit-39839	Michigan
147147	Wilson Davis	34	Wilson@gmail.com 443-958-9094	586 Nicollet Mall- Minneapolis-24832	Minnesota
148148	Lee White	32	Lee@gmail.com 484-484-999	685 Main Street-Tupelo-72732	Mississippi
149149	Jones Williams	41	Jones@gmail.com 384-855-6374	383 The Paseo-Kanas-34788	Missouri
150150	Taylor Jones	31	Taylor@gmail.com 747-383-4848	345 Last Chance Gulch-Helena-89487	MONTANA

In [14]:
```
data.to_csv(r'\Desktop\cust_transformation.csv',index=False,header=F
```

In [15]:
```
df_sale=pd.read_sql_query("SELECT * FROM cust_transformation",conn) df_sale.
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[15]:

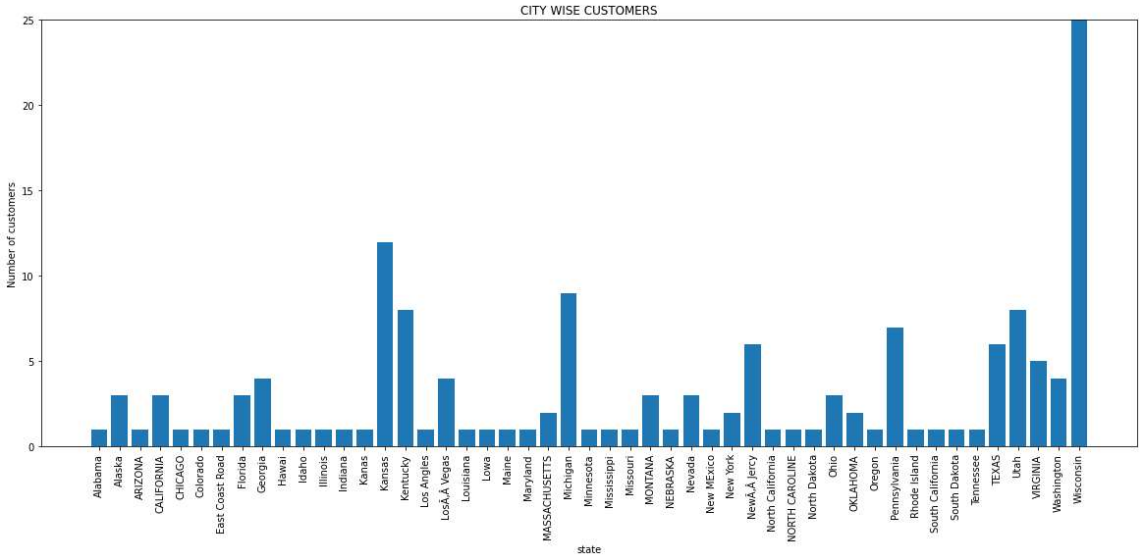
cust_id	name	age	contact	address	state
01	ADAMS ALLEN	8	adams@gmail.com 805-252-6879	3262 Euclid Avenue-Longford-67458	Kansas
12	ATKIN ANDERSON	3	atkin@gmail.com 615-246-6831	2042 Andell Road-Waynesburg-40489	Kentucky
23	THOMAS OLIVER	24	thomas@gmail.com 724-946-1278	3778 Spruce Drive-New Wilmington-16142	Pennsylvania
34	JOSHUA CHARLIE	18	joshua@gmail.com 620-328-3353	4784 Roosevelt Road-Silver Lake-53170	Wisconsin
45	JAMES CHARLIE	12	james@gmail.com NA	1718 Roosevelt Road-Cunningham-67035	Kansas

REGION WISE CUSTOMERS

```
cursor.execute("select state, count(state) from cust_transformation result =
```

```
for i in cursor:
    state.append(i[0])
    customer_count.append(i[1])

print("Name of Students = ",state)
print("Marks of Students = ",customer_count)

plt.rcParams['figure.figsize'] = [20, 8]
plt.xticks(rotation=90)
# Visulizing Data using Matplotlib
plt.bar(state,customer_count)
plt.ylim(0, 25)
plt.xlabel("state")
plt.ylabel("Number of customers")
plt.title("CITY WISE CUSTOMERS")
plt.show()
```

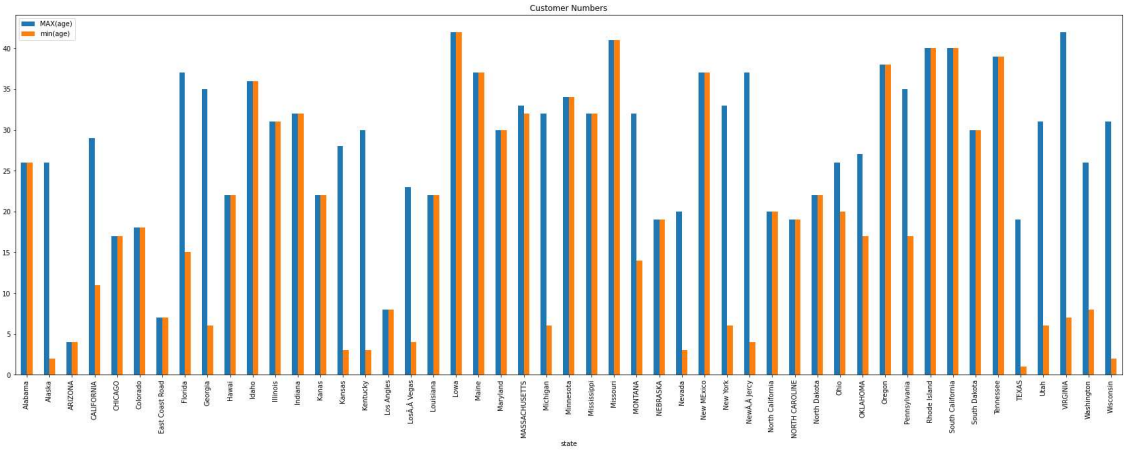
```
Name of Students = ['Alabama', 'Alaska', 'ARIZONA', 'CALIFORNIA',
'CHICAGO', 'Colorado', 'East Coast Road', 'Florida', 'Georgia', 'Ha
wai', 'Idaho', 'Illinois', 'Indiana', 'Kanas', 'Kansas', 'Kentucky
', 'Los Angles', 'LosÃ,Â\xa0Vegas', 'Louisiana', 'Lowa', 'Maine', '
Maryland', 'MASSACHUSETTS', 'Michigan', 'Minnesota', 'Mississippi',
'Missouri', 'MONTANA', 'NEBRASKA', 'Nevada', 'New MExico', 'New Yor
k', 'NewÃ,Â\xa0Jercy', 'North California', 'NORTH CAROLINE', 'North
Dakota', 'Ohio', 'OKLAHOMA', 'Oregon', 'Pennsylvania', 'Rhode Islan
d', 'South California', 'South Dakota', 'Tennessee', 'TEXAS', 'Utah
', 'VIRGINIA', 'Washington', 'Wisconsin']
Marks of Students =  [1, 3, 1, 3, 1, 1, 1, 3, 4, 1, 1, 1, 1, 1, 12,
8, 1, 4, 1, 1, 1, 1, 2, 9, 1, 1, 1, 3, 1, 3, 1, 2, 6, 1, 1, 1, 3,
2, 1, 7, 1, 1, 1, 1, 6, 8, 5, 4, 25]
```



Count of Maximum and Minimum customers by age - region wise

In [18]:
```
query="SELECT state, MAX(age), min(age) FROM cust_transformation GRO df = pd
plt.rcParams['figure.figsize'] = [30, 10] plot=df.plot.bar(title="Customer N
```

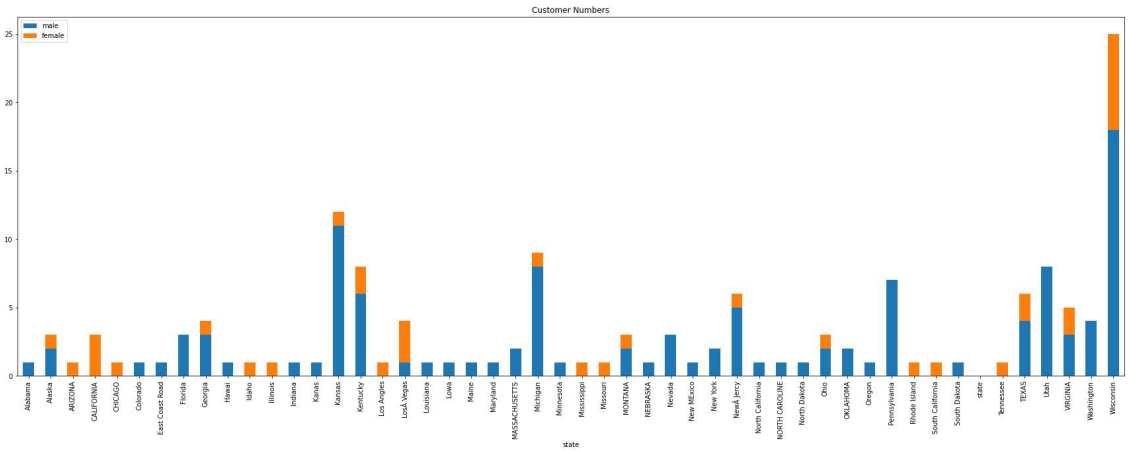
```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
```



Customer Analysis - with respect to gender

In [19]:
```
query="select state, sum(case when gender='M' then 1 else 0 end) as df = pd.
plot=df.plot.bar(title="Customer Numbers",x='state',stacked=True); fig = plc
fig.savefig("output3.png")
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```



In [ ]:

In [74]:
```
import pymysql import pandas as pd import sqlalchemy
from datetime import datetime import matplotlib.pyplot as plt import seaborn
import numpy as np
from dateutil.parser import parse
```

In [75]:
```
conn=pymysql.connect(host='localhost',port=int(3306),user='root',pas cursor=
```

In [76]:
```
df=pd.read_sql_query("SELECT * FROM sales",conn) df.head(10)
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[76]:

sale_id	cust_id	branch_id	product_id	product_price	quantity	sales_date
0S01	1	218654	P01	100	2	2022-02-15
1S02	2	289415	P02	250	3	2020-04-14
2S03	3	272682	P03	200	4	2020-11-13
3S04	4	275394	P04	180	2	2021-08-06
4S06	6	296483	P06	280	5	2021-02-27
5S07	7	214563	P07	300	6	2021-04-18
6S08	8	284566	P08	250	7	2021-03-12
7S09	9	274681	P09	200	2	2022-02-08
8S10	10	218654	P01	100	4	2021-02-07
9S100	100	274681	P35	260	5	2021-10-26

In [77]:
```
df["branch_details"]=pd.read_sql_query("select concat(b.branch_name, df["bra
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[77]:

```
0           Walmart Supercenter-Kansas
1           Walmart Supercenter-Kansas
```

In [78]:
```
df["product_details"]=pd.read_sql_query("select concat(p.product_nam df["prc
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[78]:
```
0     parle biscuit-snack
1     parle biscuit-snack
2     parle biscuit-snack
3     parle biscuit-snack
4     parle biscuit-snack

91          banana-fruits
92       tangerine-fruits
93     potato-vegetables
94     brinjal-vegetables
95      chilly-vegetables
Name: product_details, Length: 96, dtype: object
```

In [79]:
```
df["sale_amount"]=pd.read_sql_query("select product_price * quantity df["sal
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[79]:
```
0      200
1      750
2      800
3      360
4     1400

91     600
92    4080
93     800
94    3200
95    2250
Name: sale_amount, Length: 96, dtype: int64
```

In [80]:
```
df["year"]=pd.read_sql_query("SELECT date_format(sales_date,'%Y') AS df["yea
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[80]:

```
0     2022
1     2020
2     2020
3     2021
4     2021

91    2020
92    2021
```

In [81]:
```
df["month"]=pd.read_sql_query("SELECT date_format(sales_date,'%M') A df["mor
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[81]:
```
0     February
1        April
2     November
3       August
4     February

91      August
92     October
93        July
94      August
95     October
Name: month, Length: 96, dtype: object
```

In [82]:
```
data = pd.DataFrame({ 'sale_id':df["sale_id"],
'cust_id':df["cust_id"], 'branch_id':df["branch_id"], 'branch_details':df["b
'sales_month':df["month"],
})
data
```

Out[82]:

sale_id	cust_id	branch_id	branch_details	product_id	product_details	product_price	q
0S01	1	218654	Walmart Supercenter-Kansas	P01	parle biscuit-snack	100	
1S02	2	289415	Walmart Supercenter-Kansas	P02	parle biscuit-snack	250	
2S03	3	272682	Walmart Supercenter-Kansas	P03	parle biscuit-snack	200	

	sale_id	cust_id	branch_id	branch_details	product_id	product_details	product_price	q
	3S04	4	275394	Walmart Supercenter-Kansas	P04	parle biscuit-snack	180	
	4S06	6	296483	Walmart Supercenter-Kansas	P06	parle biscuit-snack	280	
	.....	...	...	...	...	...	...	
	91S94	94	284566	Crayola LLC company-Pennsylvania	P17	banana-fruits	200	
	92S95	95	274681	Crayola LLC company-Pennsylvania	P18	tangerine-fruits	1020	
	93S96	96	272682	Crayola LLC company-Pennsylvania	P12	potato-vegetables	200	
	94S97	97	275394	Crayola LLC company-Pennsylvania	P13	brinjal-vegetables	800	
	95S99	99	296483	Crayola LLC company-	P15	chilly-vegetables	450	

In [83]:
```
data.to_csv(r'\Desktop\sales_trans.csv',index=False,header=False)
```

In [84]:
```
sales_trans=pd.read_sql_query("select * from sales_transformation",c sales_t
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[84]:

	sale_id	cust_id	branch_id	branch_details	product_id	product_details	product_price	q
	0S01	1	218654	Walmart Supercenter-Kansas	P01	parle biscuit-snack	100	
	1S02	2	289415	Walmart Supercenter-Kansas	P02	parle biscuit-snack	250	
	2S03	3	272682	Walmart Supercenter-Kansas	P03	parle biscuit-snack	200	
	3S04	4	275394	Walmart Supercenter-Kansas	P04	parle biscuit-snack	180	
	4S06	6	296483	Walmart Supercenter-Kansas	P06	parle biscuit-snack	280	
	.....	...	...	...	...	...	...	
	91S94	94	284566	Crayola LLC				

sale_id	cust_id	branch_id	branch_details	product_id	product_details	product_price	q
92S95	95	274681	Crayola LLC company-Pennsylvan	P18	tangerine-fruits	1020	
93S96	96	272682	Crayola LLC company-Pennsylvan	P12	potato-vegetables	200	
94S97	97	275394	Crayola LLC company-Pennsylvan	P13	brinjal-vegetables	800	
95S99	99	296483	Crayola LLC company-Pennsylvan	P15	chilly-vegetables	450	

In [85]:
```
sale=pd.read_sql_query("select branch_details, count(branch_details)
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

In [86]:
```
sale
```

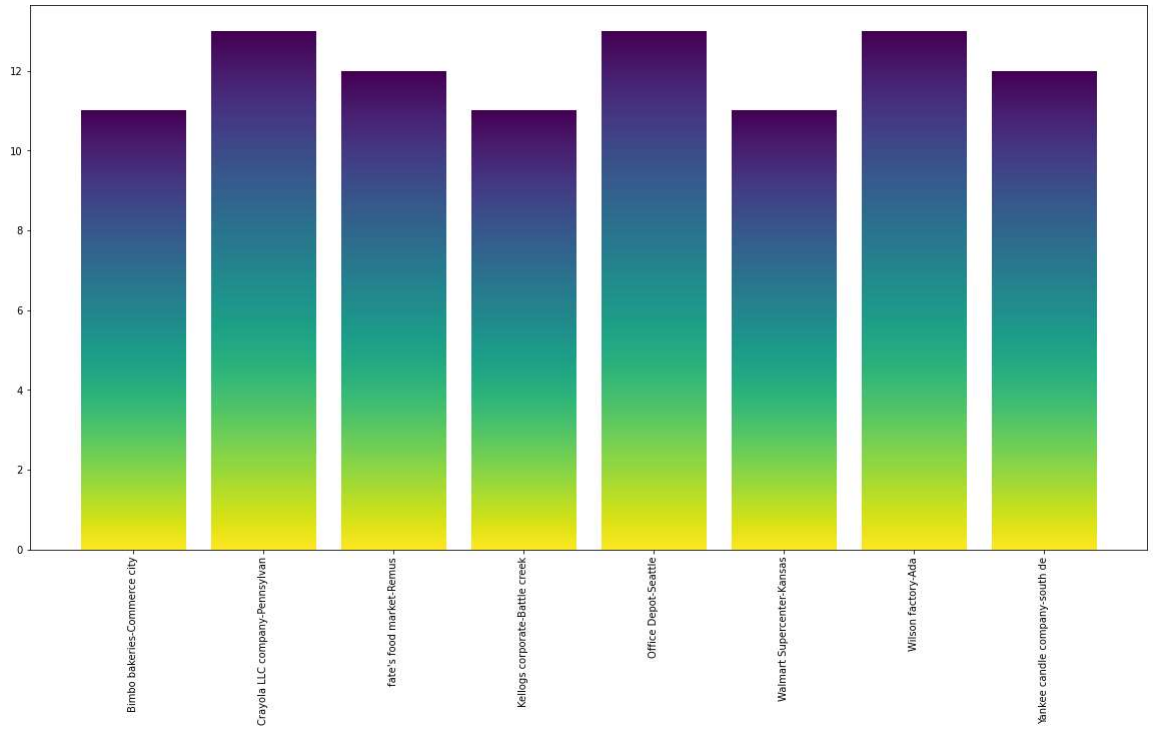
Out[86]:

branch_details	count(branch_details)
0Bimbo bakeries-Commerce city	11
1Crayola LLC company-Pennsylvan	13
2fate's food market-Remus	12
3Kellogs corporate-Battle creek	11
4Office Depot-Seattle	13
5Walmart Supercenter-Kansas	11
6Wilson factory-Ada	13
7Yankee candle company-south de	12

Branch wise Sale

In [104]:
```
cursor.execute("select branch_details, count(branch_details) from sa result
branch= [] sale_count= []

for i in cursor: branch.append(i[0]) sale_count.append(i[1])

fig, ax = plt.subplots()

bar = ax.bar(branch,sale_count) plt.xticks(rotation=90)
def gradientbars(bars):
grad = np.atleast_2d(np.linspace(0,1,256)).T ax = bars[0].axes
```

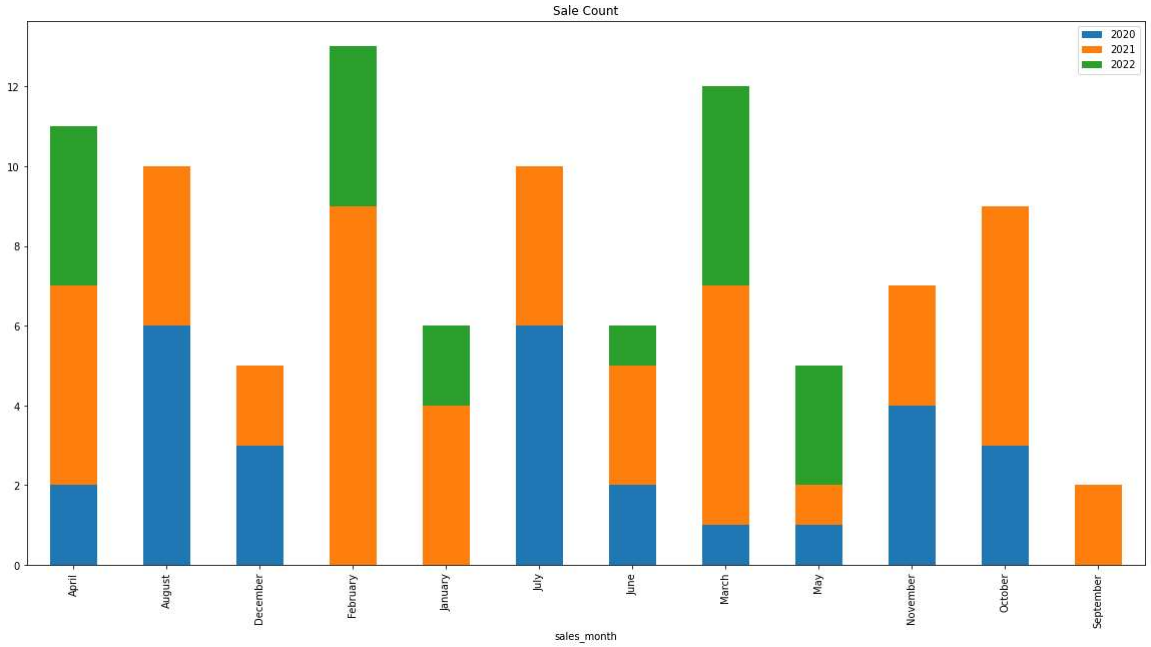
```
lim = ax.get_xlim()+ax.get_ylim()
for bar in bars: bar.set_zorder(1) bar.set_facecolor("none") x,y = bar.get_x
w, h = bar.get_width(), bar.get_height()
ax.imshow(grad, extent=[x,x+w,y,y+h], aspect="auto", zorder= ax.axis(lim)

gradientbars(bar) plt.rcParams['figure.figsize'] = [20, 10]

plt.show()
```



Year wise sale -WRT- Month

In [105]:
```
query="select sales_month, sum(case when sales_year='2020' then 1 el df = pd
plot=df.plot.bar(title="Sale Count",x='sales_month',stacked=True); plt.rcPar
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```



In [96]:
```
year=pd.read_sql_query("select sales_year, count(sales_year) from sa year
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

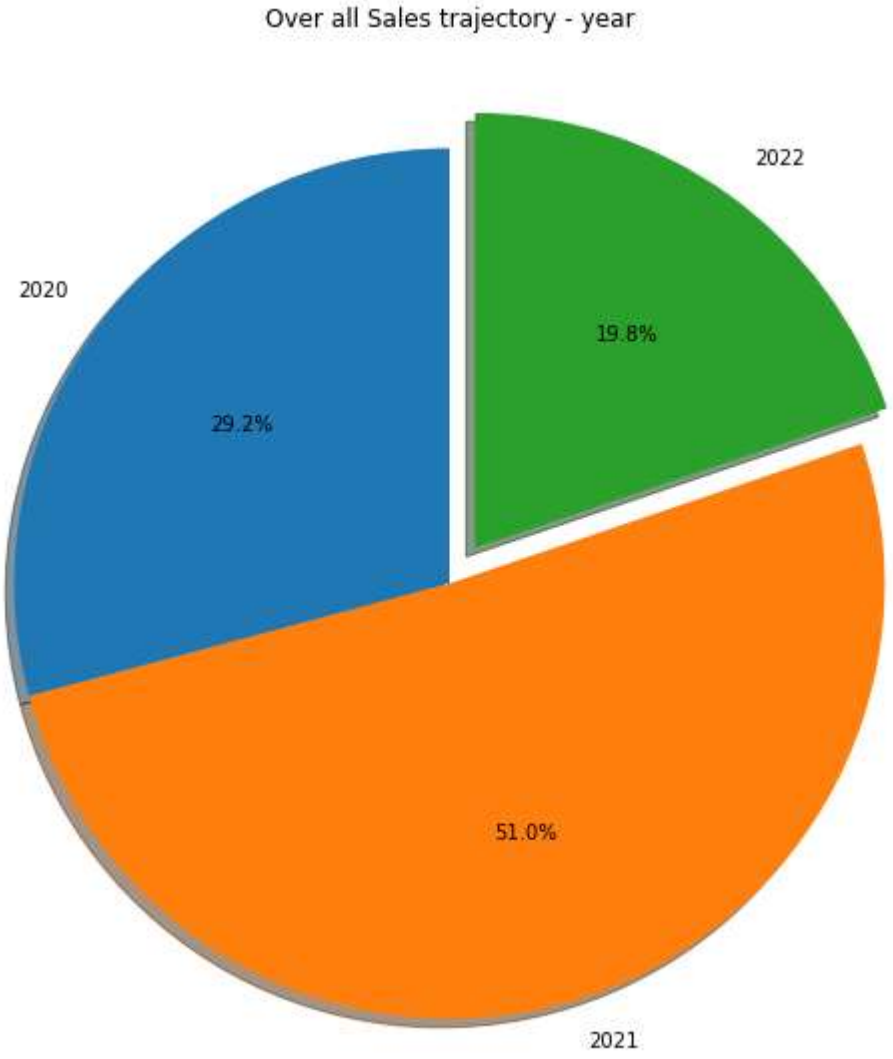
Out[96]:

sales_year	count(sales_year)
02020	28
12021	49
22022	19

Sale trajectory - over the years

In [101]:
```
classes = '2020', '2021', '2022'
query="select count(sales_year) from sales_transformation group by s cursor.
result = cursor.fetchall() final_result = [i[0] for i in result]

explode = (0, 0, 0.1)
# only "explode" the 3rd slice (i.e. '2022')
```

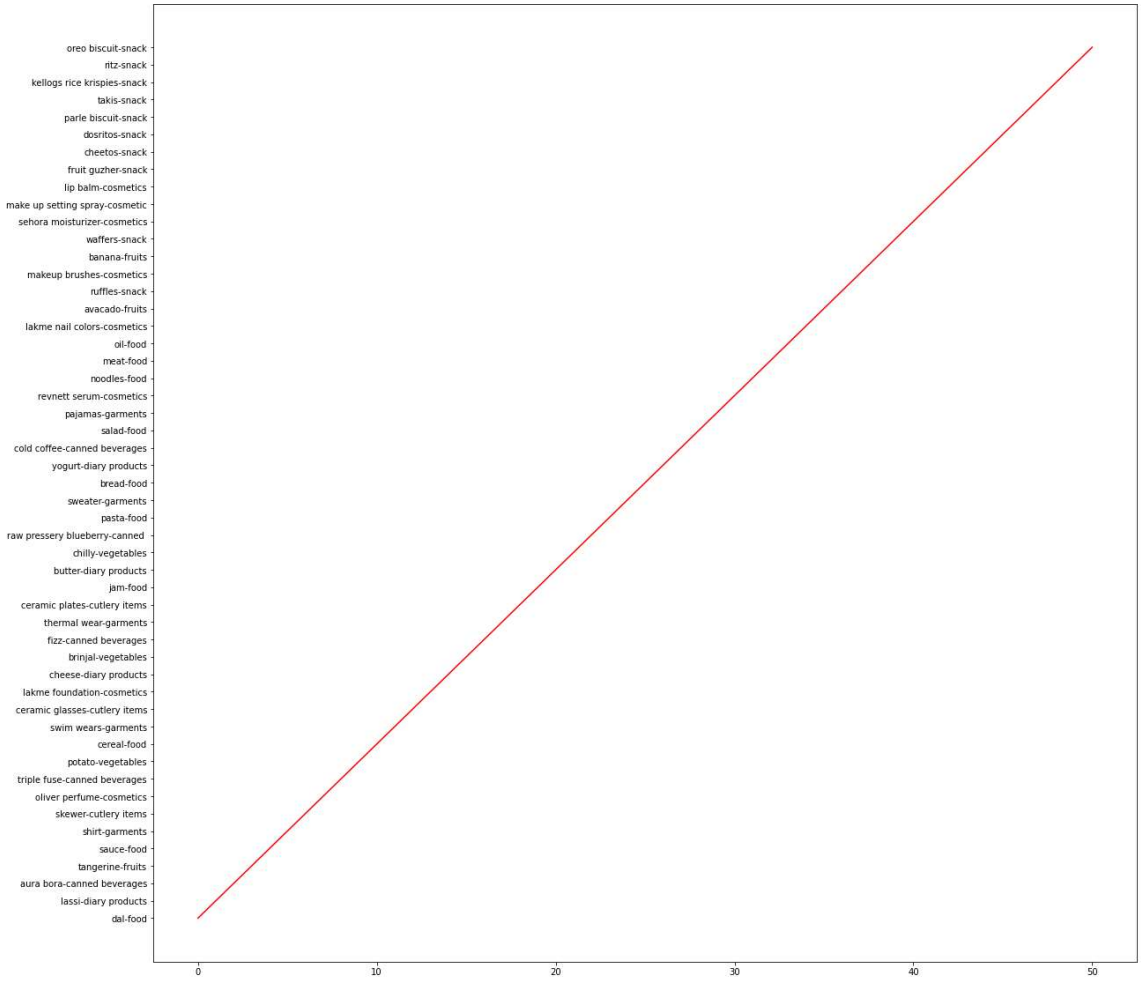
```
plt.pie(final_result, explode=explode, autopct='%1.1f%%', labels=classes,sha

plt.title("Over all Sales trajectory - year") plt.rcParams['figure.figsize']
```



Most sold vs Least Sold & everything in between

In [107]:
```
df=pd.read_sql_query("select product_details, count(sale_id) from sa plt.plo
plt.rcParams['figure.figsize'] = [20, 20] plt.show()
```

```
/home/ubh01/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:
761: UserWarning: pandas only support SQLAlchemy connectable(engine
/connection) ordatabase string URI or sqlite3 DBAPI2 connectionothe
r DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```



In [ ]:

8. Further Enhancements/Recommendations

This project has a very vast scope in the future and even present for all product based companies. We developed this project for US based Retail Store but can be generalised for all other retail store, etc. There are various use cases that can be achieved by this project. Some of the future scopes are mentioned below:

- From this data to learn about the way to keep customers returning to your shop is by offering new and exciting products. When adding new products or expanding product lines, keep in mind that not only should there be a demand for the item, but it must also
  be profitable and something you enjoy selling.
- The companies can develop algorithms in such a way that only preferred products reaches the correct audience which can result in popularity between the customers.

9. References/Bibliography

- DRAW.IO
- Big Data Ethics
- Project Jupyter
- Data Analysis Using python
- The Ultimate Hands-on Hadoop: Tame you Big Data!

GITHUB LINK

https://github.com/SwethaMuthuvel/US-based-retail-sale-analysis