Acknowledgment

I would like to take this opportunity to express my heartfelt gratitude to all those who have contributed to the successful completion of my Class 12 IP project, "Sales Insight."

First and foremost, I would like to extend my deepest appreciation to my IP teacher Vaibhav Sharma for their unwavering support, guidance, and expertise throughout this project. Their invaluable insights, constructive feedback, and dedication to my learning have been instrumental in shaping this project into what it is today.

I would also like to thank my parents and family for their constant encouragement and understanding during this project. Their patience and belief in my abilities have been a tremendous source of motivation.

I extend my thanks to my classmates and friends who provided valuable input and encouragement along the way. Your discussions and brainstorming sessions were immensely helpful in refining the project's ideas and concepts.

Furthermore, I would like to acknowledge the support and cooperation received from Sanskar The Co-Educational school, which provided the necessary resources and infrastructure for the successful execution of this project.

This project has been an enriching experience, and I am grateful for the opportunity to work on it. The knowledge and skills I have gained during this project will undoubtedly prove beneficial in my future endeavors.

Sincerely,

Omansh Arora XII sci

Certificate

This is to certify that Omansh Arora of Class 12 sci has successfully completed the project titled "Sales Insight" of Subject: Informatics Practices under the guidance of Vaibhav Sharma on . This project demonstrates exceptional competence in the field of Informatics Practices and showcases a deep understanding of the concepts related to sales data analysis.

This project reflects the dedication and hard work put forth by the student in mastering the principles of Informatics Practices, data handling, and analysis. It demonstrates their ability to apply these skills to real-world scenarios effectively.

We commend the student for their commitment to excellence in the subject of Informatics Practices and wish them continued success in their academic and professional endeavors.

The principal

Chhaya Agarwal

Sanskar The Co-Educational school

_____

Signature of Principal

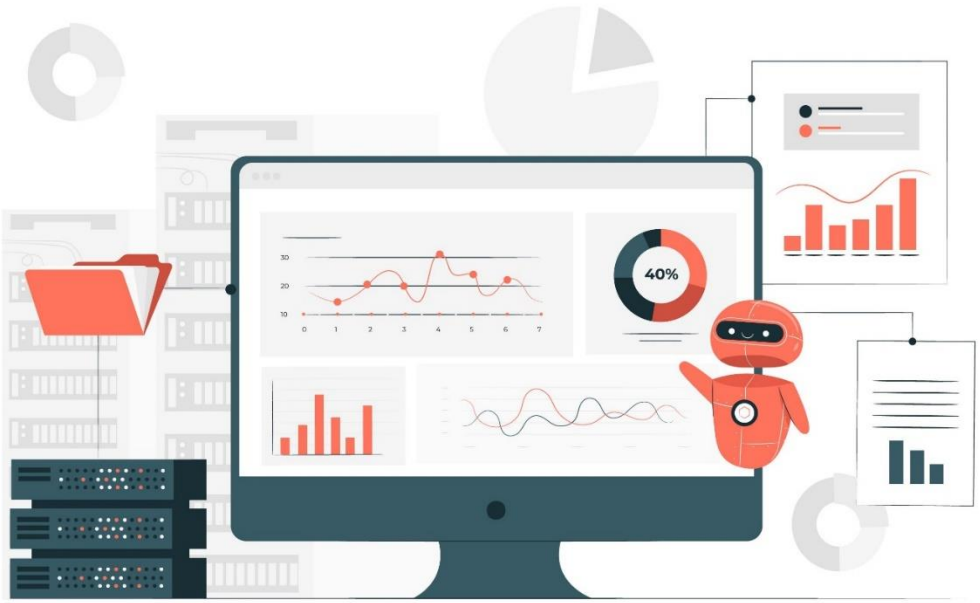
_____ _____

Signature of external examiner Signature Subject in charge

INDEX

Sales Analysis Project

Problem Statement



XYZ hardware is a company in India which supplies computer hardware and peripheral devices across India only. They have many stores across India such as surge stores, Nomad stores etc. The head office of the company is situated in Delhi.

The sales manager of the company is facing many challenges. He is facing issues in tracking sales in a dynamically growing market. He is having issues with the insights of his business.

In order to this he has some of the regional managers in North, south and central India working for the company. So, he calls them and asks about the insights he wants to know. They told him about the sales in the last quarter and the growth in that quarter.

So, the problem is that the conversations that are happening are verbal. Hence, the regional managers are sugarcoating the facts, and the manager of the company does not get a clear picture of the facts. Even after knowing that the sales are declining, he cannot do anything because he does not have a clear picture of the sales.
Asking for the records the regional manager provides him with excel files. But by this he cannot figure out Market wise Profit, Total Profit, Avg selling point etc.

All what the manager wants is a view of the weakest area the company needs to focus on to increase the sales and improvise the decline. He is interested in simple, understandable and digestive insight. So, he is more interested in a dashboard which he can go and look at the real data because data speaks the truth. All he wants is a simple data visualization tool which he can access on a daily basis.

.

Hence, by using such tools and technology one can make data-driven decisions which help to increase the sales of the company.
So, in this project we will help a company make its own sales-related dashboard using Python programming, pandas and other libraries and other Technology

Project planning using AIMS grid

AIMS grid: is a lateral leadership tool that helps define and explain goals. It contains four dimensions:

1) Purpose (what to do exactly)

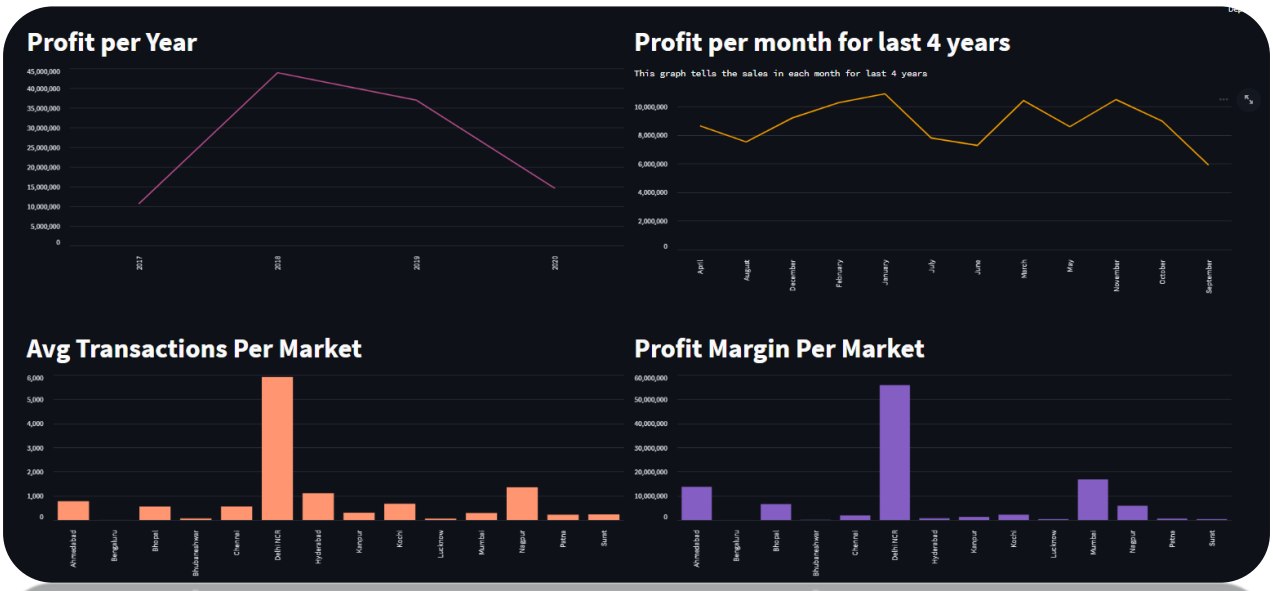
2) Stack holders (who will be involved)

3) End result (what do you want to achieve)

4) Success criteria (cost optimization and time save)

In our case the end result will be the dashboard created and success criteria will bump up the sales using cost optimization and save the time of the manager of the company.

The AIMS grid is a project management tool that can help project managers and team leaders. It can help clarify a task and keep the process simple. The AIMS grid presents a project in a concise manner



AIMS GRID

PURPOSE	STAKEHOLDERS
To unlock sales insights that are not visible before for the sales team for decision support & automate them to reduced manual time spent in data gathering.	• Sales Director • Marketing Team • Customer Service Team • Data & Analytics Team • IT
END RESULT	**SUCCESS CRITERIA**
An automated dashboard providing quick & latest sights in order to support data driven decision making.	• Dashboard uncovering sales order insights with latest data available. • Sales team able to take better decisions & prove 10% cost saving of total spend • Sales Analysts stop data gathering manually in order to save 20% of business time and reinvest it value added activity.

PROJECT MODLE

The **Waterfall Model** is a traditional project management methodology that is well-suited for projects with well-defined and stable requirements, such as a Sales Insight Project. This model provides a structured and sequential approach to project development. Here is an overview of the Waterfall Model for a Sales Insight Project:

Requirements Gathering and Analysis:

Scope Definition: The project's scope, objectives, and deliverables are clearly defined. In the context of a Sales Insight Project, this phase involves understanding the specific sales data sources, the business objectives, and the key performance indicators (KPIs) that need to be tracked.

Requirements Gathering: Collect detailed requirements from stakeholders, including sales managers, analysts, and IT personnel. These requirements may involve data sources, data formats, report formats, and user access levels.

System Design:

Architectural Design: Develop a detailed system architecture, which may include decisions on the database structure, data processing pipeline, and data visualization tools.

Database Design: Create a database schema that accommodates the data required for sales analysis, ensuring data integrity and security.

User Interface Design: Design the user interface for the Sales Insight application, taking into consideration the specific needs and preferences of end-users.

Implementation:

Data Acquisition: Implement the processes to collect and ingest data from various sources, such as CRM systems, sales databases, and external market data sources.

Data Processing: Develop the data processing logic to clean, transform, and prepare the data for analysis.

Application Development: Build the Sales Insight application based on the user interface design, integrating it with the data processing pipeline and the database.

Testing:

Unit Testing: Test individual components and functions to ensure they work correctly.

Integration Testing: Verify that different system components, including data acquisition, data processing, and the user interface, work together as expected.

User Acceptance Testing (UAT): Involve end-users to validate the application's functionality and usability against their requirements.

Deployment:

Installation: Deploy the Sales Insight application on the designated server or cloud environment.

Training: Provide training to end-users and administrators on how to use the application effectively.

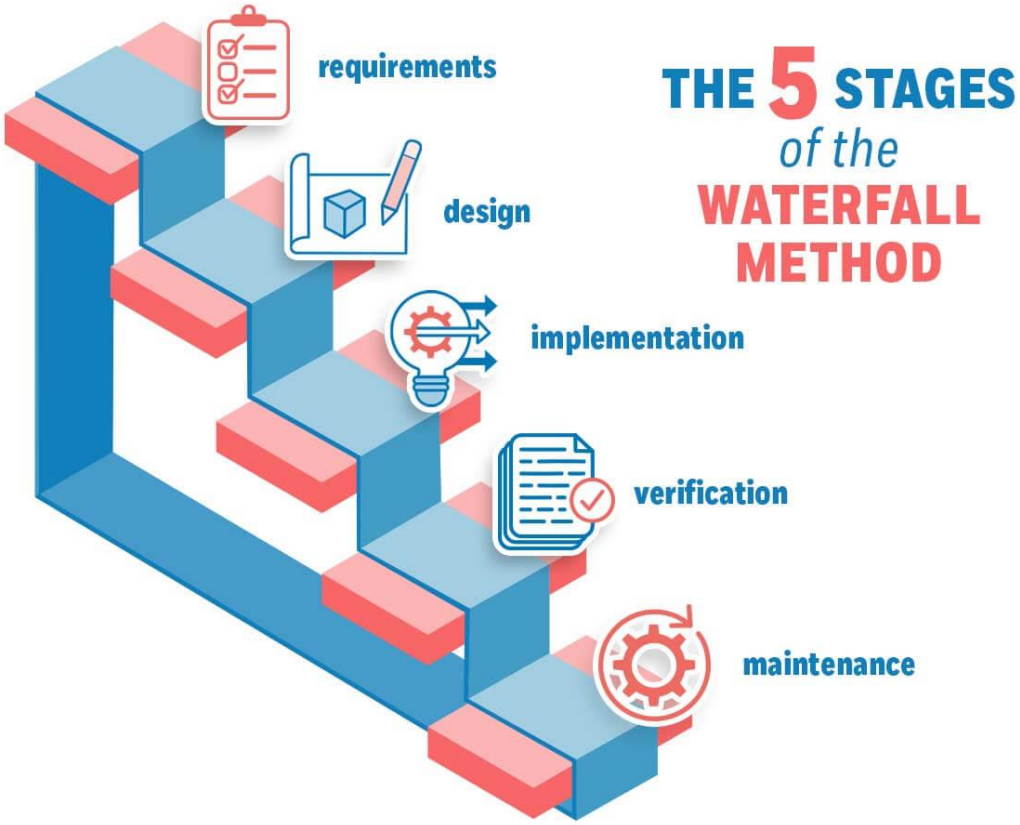
Maintenance and Support:

Establish ongoing maintenance and support procedures to address issues, apply updates, and make enhancements as necessary.

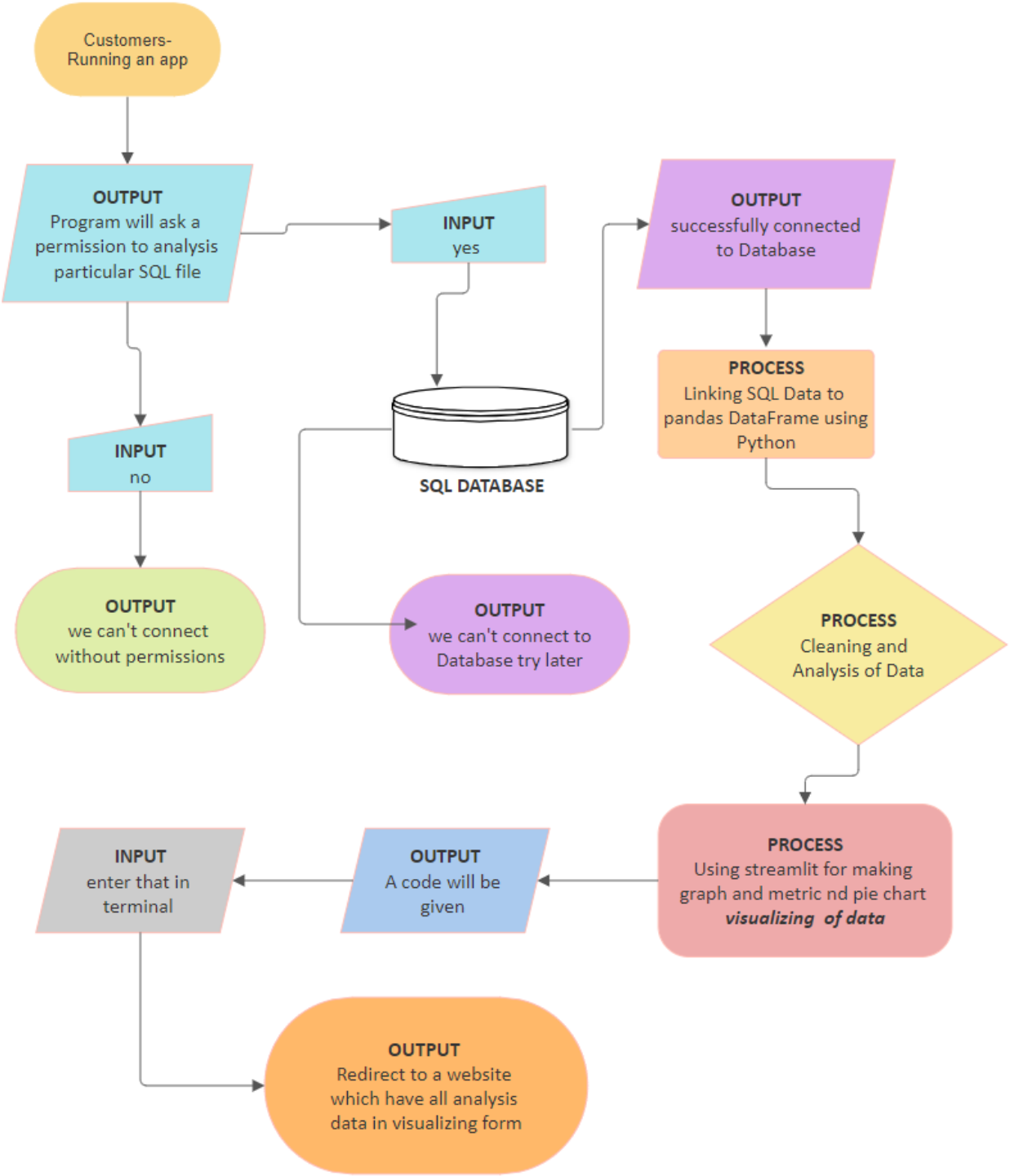
The Waterfall Model is known for its rigidity and lack of flexibility, as it follows a linear and sequential approach. It is best suited for

projects where the requirements are well-defined and unlikely to change significantly. If requirements are expected to evolve or if there's a need for iterative development and quick adjustments, an Agile methodology may be a better choice.

For a Sales Insight Project, the Waterfall Model can be effective when the project's scope and objectives are stable and well-understood.



Data Flow Diagram



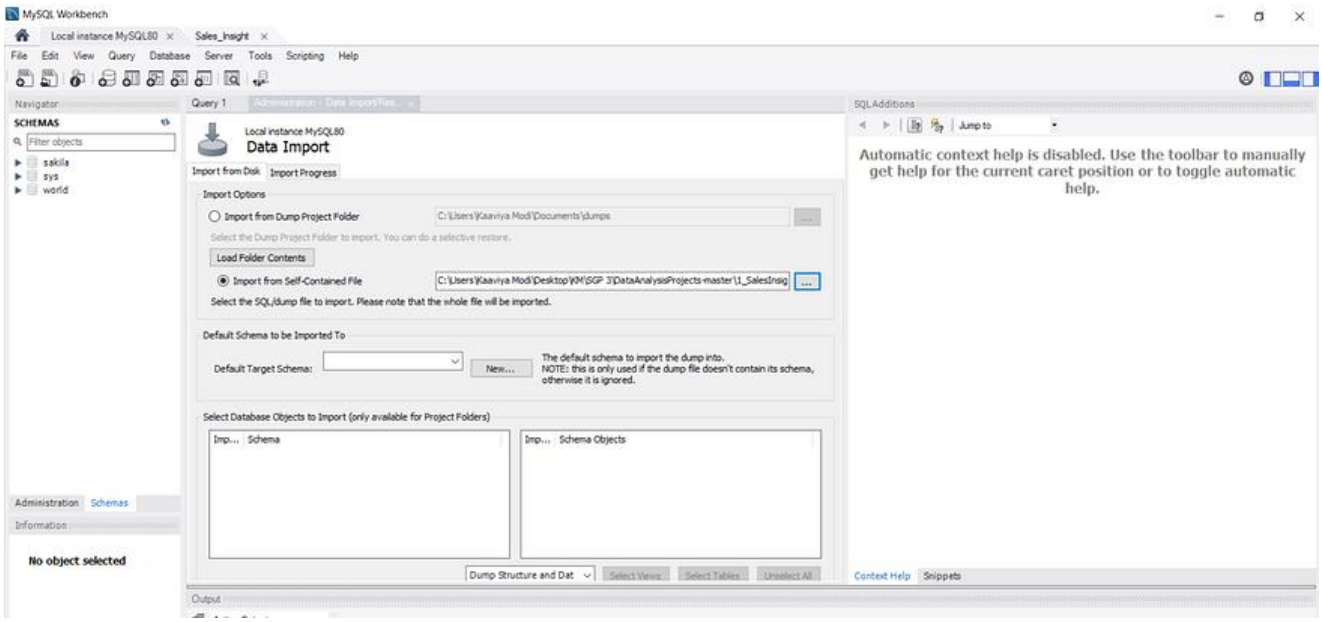
How will the company work

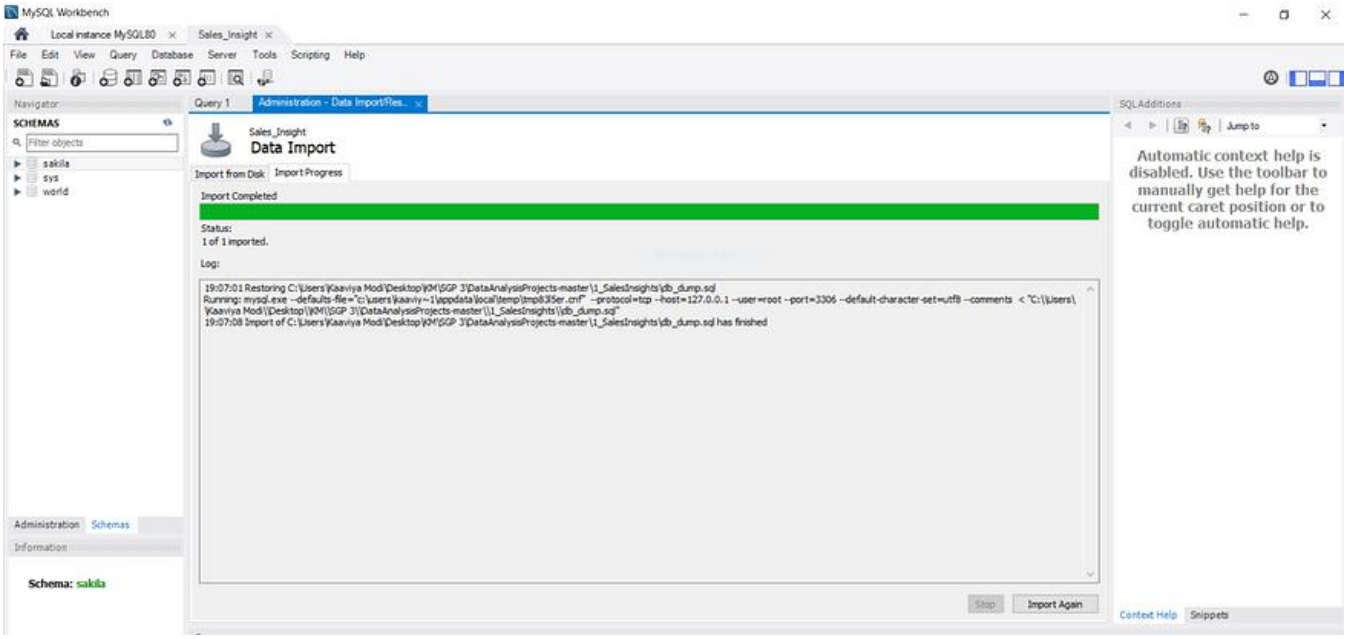
There is a team of software engineers (falcons) which owns sales management system. The records of this system are stored in MySQL database.
Company hires a team of Data Analyst (Data masters) reaches out to the software engineers to get an access to data base which they can use to create the dashboard in pandas and other libraries

In this same manner our project is going to be executed. We are going to fetch the data from the database and then we are going to transform and load the data onto the pandas to build the dashboard.

Reading and Optimizing SQL

Step 1: Importing Data to MySQL workbench

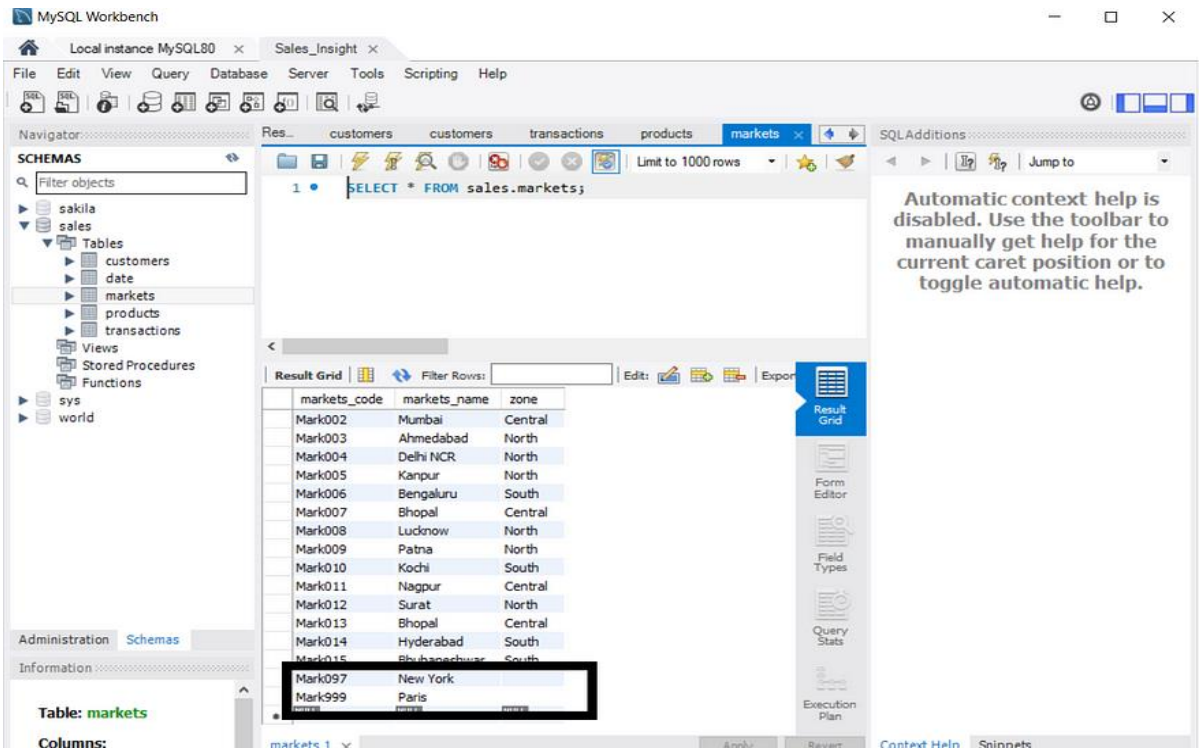


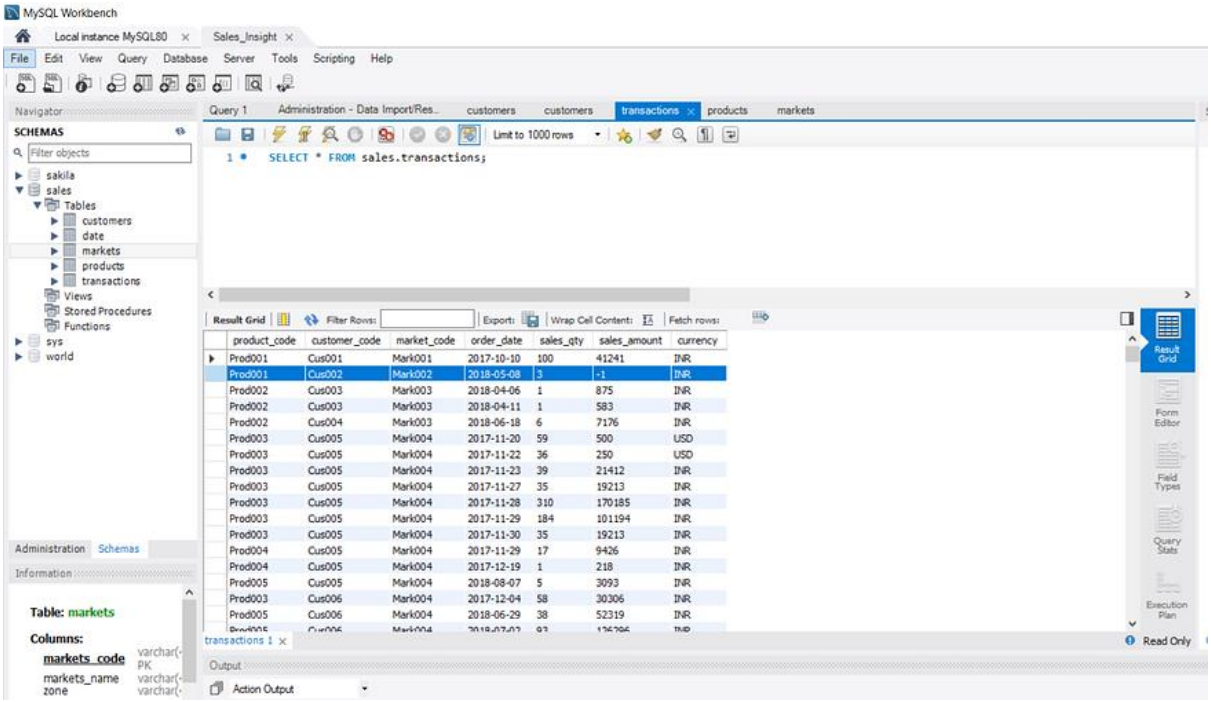


The import of data is done from an already existing MySQL file. This file has to be loaded into MySQL workbench for further data analysis.

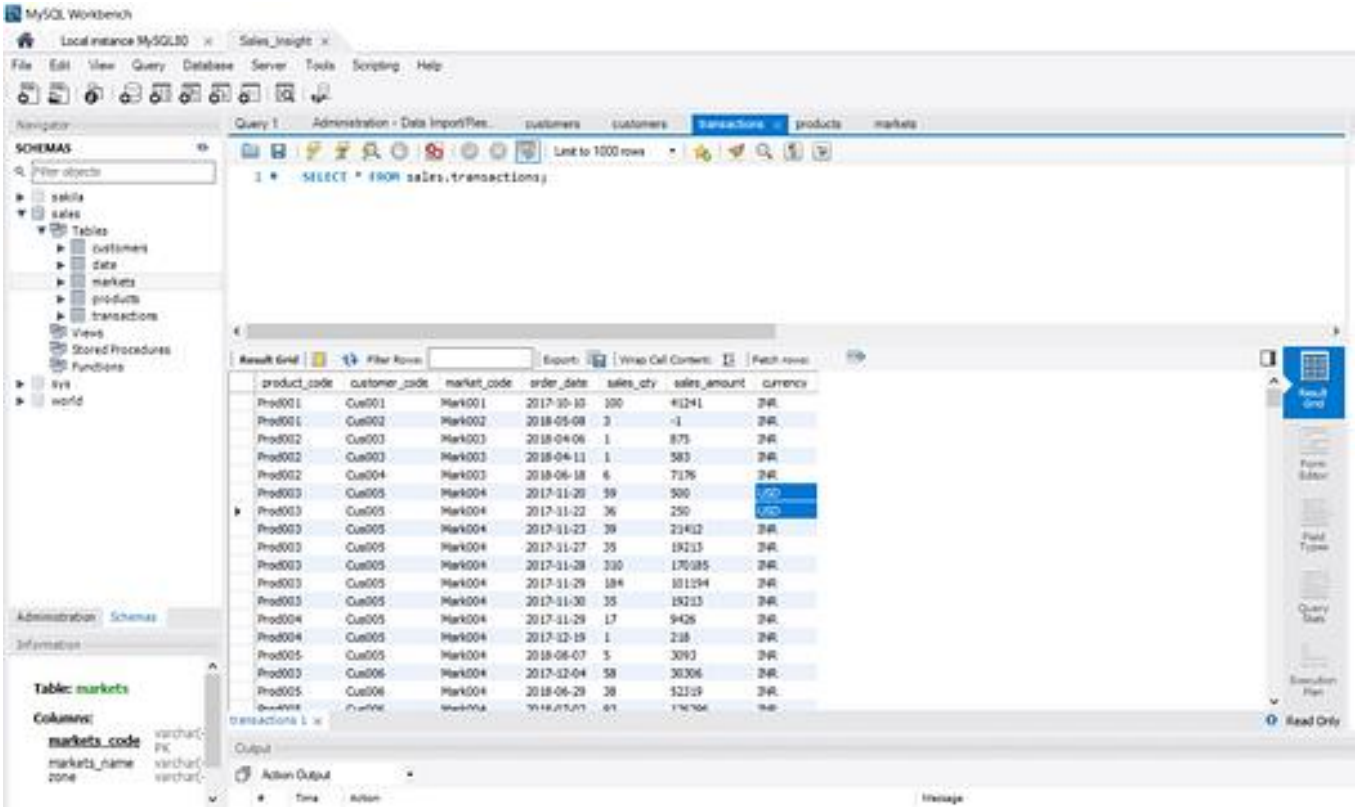
The following images show that the import is a success.

Here, we can see that the table market contains certain values which are incorrect. XYZ hardware company works only in India but there are some records of different non-existing cities in India.





Here, we can see that table transactions contain a certain negative value in amount which is not possible.



Here, you can see that certain transactions are in USD. Hence, filtration of that is also needed by converting into INR

All the data cleaning will be done on pandas as it is possible here too, but it will be more easy, future friendly and scalable

Now all the work will done on pandas

Source Code

```
#source code of main program which will be responsible for running nd
analysing data


#importing all the necessary libraies

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import streamlit as st
import mysql.connector as ms
import math
import warnings
from decimal import Decimal
import seaborn as sns
warnings.filterwarnings("ignore")

#asking user to eneter if he want to analysis default file or new one


which_file=input('Do u want to analysis the default file(dumb_sql_2020).
choose the answer as per question and answer in the option\ni)Y\nii)N\n')
if which_file=='Y':
    print('we are heading towords to analysis dumb_sql_2020')

 mycom=ms.connect(host='localhost',user='root',passwd='000001',database='s
ales')
    if mycom.is_connected():
        print('wow! we successfully connected to dumb_sql_2020')

#after successfully connected to our sql file we mare linking sql with
python by making sql tables in database using pandas
```

```
slecting_tables_1='select * from customers;'
slecting_tables_2='select * from date;'
slecting_tables_3='select * from markets;'
slecting_tables_4='select * from products;'
slecting_tables_5='select * from transactions;'
customers=pd.read_sql(slecting_tables_1,mycom)
date=pd.read_sql(slecting_tables_2,mycom)
markets=pd.read_sql(slecting_tables_3,mycom)
products=pd.read_sql(slecting_tables_4,mycom)
transactions=pd.read_sql(slecting_tables_5,mycom)
#successfully made 5 database of sql tables for our data analysis
#print(transactions.info(),'\n',customers.info(),'\n',
date.info(),'\n',markets.info(),'\n',products.info())
# after doing df.info() we get to know there are data column in
object type so will change it into datetime type
if transactions['order_date'].dtypes == 'object':
    transactions['order_date']=pd.to_datetime(transactions[
    'order_date'])
if date['date'].dtype=='object':
    date['date']=pd.to_datetime(date['date'])
#print(transactions,'\n',date)
#print(transactions.isnull().sum())
#no null value

transactions=transactions[transactions['profit_margin']>=0]
transactions=transactions[transactions[
'profit_margin_percentage']>=0]
transactions.loc[transactions['currency']=='USD',['sales_amount']]
=transactions['sales_amount']*80
#done with cleaning now moving towards analysis
markets.rename(columns=
{'markets_code':'market_code'},inplace=True)
#no markets_code named column in transcation df so to merge them
we jhave to make it linkable
transactions_for_market =pd.merge(transactions,
markets[['market_code','markets_name']],on='market_code',
how='left')
date.rename(columns=
{'date':'order_date','date_yy_mmm':'date_mm'},inplace=True)

transactions_by_date=pd.merge(transactions,date[[
'order_date','year','month_name']] ,on='order_date',how='left')
```

```
transactions_for_market_grp_avg=(transactions_for_market.groupby(
'markets_name')).mean()

transactions_for_market_grp_total=(transactions_for_market.groupby(
'markets_name')).sum()

transactions_by_date_grp_avg_year=(transactions_by_date.groupby(
'year')).mean()

transactions_by_date_grp_avg_month=(transactions_by_date.groupby(
'month_name')).mean()

transactions_by_date_grp_year_total=(transactions_by_date.groupby(
'year')).sum()

transactions_by_date_grp_avg_month_for_graph=
(transactions_by_date.groupby('month_name')).sum()
#new_order = ['January', 'February', 'March', 'April', 'May', 'June',
'July', 'August', 'September', 'October', 'November', 'December']
transactions_by_date_grp_year_total_for_GRAPH=
(transactions_by_date.groupby('year')).sum()
transactions_by_date_grp_year_total.reset_index(inplace=True)
transactions_by_date_grp_year_total.index
=transactions_by_date_grp_year_total.index.astype(str)
transactions_for_market_cus=(transactions_for_market.groupby(
'market_code')).size()
customers_size=transactions_for_market.groupby('customer_code').size()

customers_count_in_market = transactions_for_market.groupby(
'markets_name')['customer_code'].count()
grouped_transactions = transactions_for_market.groupby(['markets_name',
'customer_code'])
customer_transactions = grouped_transactions.size().reset_index(
name='transactions')

average_transactions_per_customer_in_market
=customer_transactions.groupby('markets_name')['transactions'].mean()
markets_with_unique_customers =
transactions_for_market.groupby('markets_name' )
['customer_code'].nunique()
```

```
total_revenue_per_market =
transactions_for_market_grp_total['sales_amount']
# Convert the pandas Series to a float before performing the
division
#normal_total_cost_spent_per_market =
(total_cost_spent_per_market.apply(lambda x: x / math.pow(10,
math.floor(math.log10(x)))))*10
#print(normal_total_cost_spent_per_market)
#****(IMP) the above answer is in the lahks

total_profit_margin_per_market=transactions_for_market_grp_total
['profit_margin']
#print(total_profit_margin_per_market)
def readable_number_converter(num):
    if num >= 10**6:
        return f"₹{num / 10**7:.2f}Crore"
    elif num >= 10**3:
        return f"₹{num / 10**5:.2f}Lakh"
    else:
        return str(num)

#if case it show 0 in front of the number is because the number
is too large to be represented as a float64. In Python, the
float64 data type can represent numbers with approximately 15
decimal digits of precision.

#The number ₹52.5458Crore is too large to be represented as a
float64, so Python is rounding it to the nearest number that can
be represented. In this case, Python is rounding it to 0.

#To avoid this issue, we can use the `Decimal` data type from
the`decimal` module, which can represent numbers with arbitrary
precision.

total_profit_margin=transactions_for_market_grp_total[
'profit_margin'].sum()

total_revenue_per_market=transactions_for_market_grp_total[
'sales_amount']
```

```
total_profit_margin=
(pd.Series(total_profit_margin).apply(readable_number_converter
)).to_string(index=False)

total_revenue=transactions_for_market_grp_total[
'sales_amount'].sum()

#The error message we are seeing is because you're trying to
call the `apply()` method on a numpy float64 object, which
doesn't have that method.
#To fix this, you can convert the numpy float64 object to a
pandas Series object before applying the
`readable_number_converter()` function.
total_revenue=(pd.Series(total_revenue).apply(
readable_number_converter)).to_string(index=False)

avg_revenue_per_month=transactions_by_date_grp_avg_month[
'sales_amount'].mean()
# Assuming transactions_by_date_grp_avg_month is a DataFrame
with a DatetimeIndex
#transactions_by_date_grp_avg_month.index =
pd.to_datetime(transactions_by_date_grp_avg_month.index)
#transactions_by_date_grp_avg_year =
transactions_by_date_grp_avg_year.assign( year=lambda x:
x.index.year)
sales_amount_2019 = transactions_by_date_grp_year_total[
transactions_by_date_grp_year_total['year'] == 2019]
['sales_amount'].values[0]
sales_amount_2020 = transactions_by_date_grp_year_total[
transactions_by_date_grp_year_total['year'] == 2020]
['sales_amount'].values[0]
percentage_increase = ((((sales_amount_2020 - sales_amount_2019)
/ sales_amount_2019) * 100).round(2))
avg_revenue_per_month_1 = (avg_revenue_per_month).round(2)

best_place=total_profit_margin_per_market.idxmax()
```

```
total_profit_margin_2019 = transactions_by_date_grp_year_total[
transactions_by_date_grp_year_total['year'] == 2019]
['profit_margin'].sum()
total_profit_margin_2020 = transactions_by_date_grp_year_total[
transactions_by_date_grp_year_total['year'] == 2020]
['profit_margin'].sum()

# Calculate the percentage decrease in total profit margin from
2019 to 2020
percentage_decrease = (((total_profit_margin_2020 -
total_profit_margin_2019) / total_profit_margin_2019) *
100).round(2)
#we are taking sample size of 2020 and 2019 data only
# Convert the percentage decrease to a readable number
readable_percentage_decrease =
readable_number_converter(percentage_decrease)
transactions_by_date_grp_year_total_for_GRAPH.index=
transactions_by_date_grp_year_total_for_GRAPH.index.astype(str)
#avg_decrease =
transactions_by_date_grp_avg_year.groupby('year')
['customers'].agg([ 'count', 'sum']).reset_index()
#avg_decrease['avg_decrease'] = avg_decrease['sum'] /
avg_decrease['count']
#The `.values[0]` is used to extract the value from a pandas
Series. In this case, it is used to extract the sales amount for
the year 2019 and 2020 from the
`transactions_by_date_grp_avg_year` DataFrame.

#Here's a step-by-step breakdown of the code:

#1`transactions_by_date_grp_avg_year[
transactions_by_date_grp_avg_year['year'] == 2019]
['sales_amount']` selects the sales amount for the year 2019.

#2.values` converts the pandas Series into a numpy array.

#3.[0]` extracts the first element from the numpy array, which
is the sales amount for the year 2019.
```

```
#The same logic is applied for the year 2020.

#So, `sales_amount_2019` and `sales_amount_2020` are the sales
amounts for the years 2019 and 2020, respectively.

#The `percentage_increase` is calculated by subtracting the
sales amount for 2019 from the sales amount for 2020, dividing
the result by the sales amount for 2019, and multiplying the
result by 100 to get the percentage increase.
own_brand_str=str('Own Brand\r')
distribution_str=str("Distribution\r")

# Count the number of products in each category
own_brand_count = products.loc[products['product_type'] ==
own_brand_str].count()
distribution_count = products.loc[products['product_type'] ==
distribution_str].count()
# Create a dashboard with Streamlit

print('copy thiscommand nd paste it here(vs terminal) or in cmd
after defining path ')
st. set_page_config(layout="wide")
```

```
    #css styling for better looks
        st.markdown("""
        <style>
        .big-font {
            font-size:150px !important;
            text-decoration: underline;
            text-align: center;
            display: inline-block;
            background-color: #212A3E;
            border: 10px solid #394867;
            padding: 2.5% 17% 2.5% 17%;
            border-radius: 18px;
            color:#F1F6F9

        }
        </style>
        """, unsafe_allow_html=True)

        st.markdown('<p class="big-font"<b>Sales Analysis</b></p>',
        unsafe_allow_html=True)


        st.write('---')
        st.container()
        st.empty()
        st.markdown(
            """
        <style>
        div[data-testid="metric-container"] > label[data-
        testid="stMetricLabel"] > div {
        overflow-wrap: break-word;
        white-space: break-spaces;
        text-align: center;
        background: #012445;
        border: 2.5px solid #CCCCCC;
            padding: 5% 7% 5% 10%;
            border-radius: 7px;

        }
```

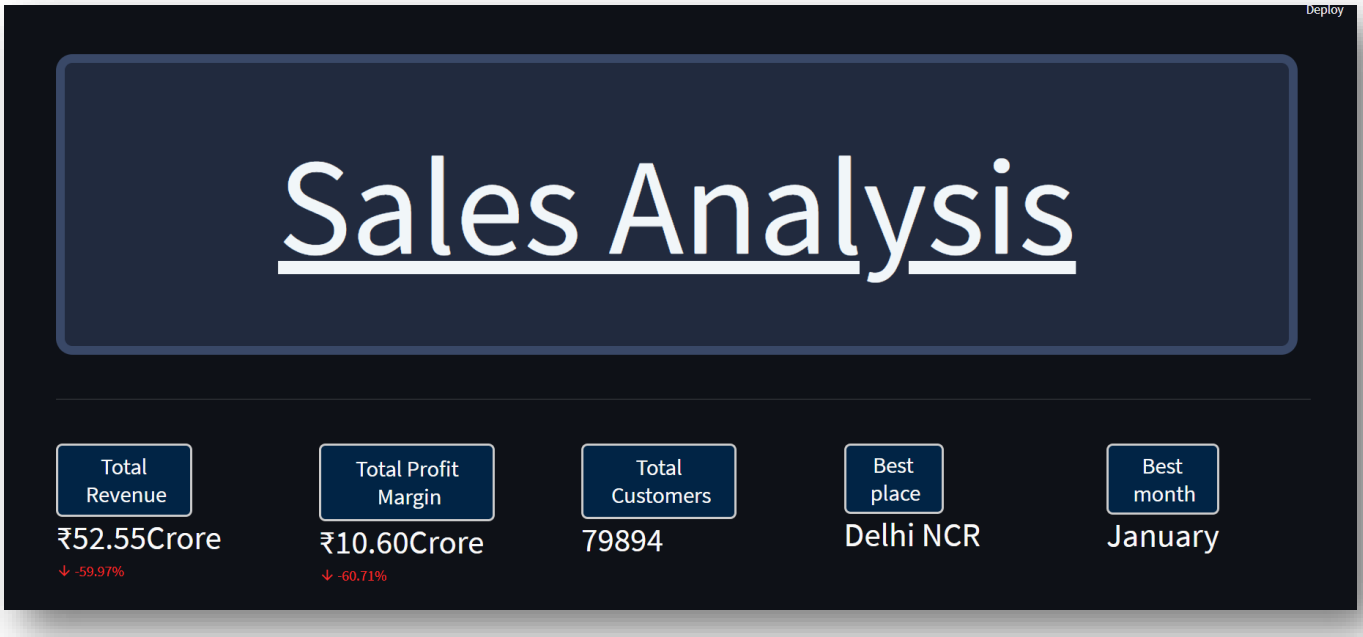
```
        div[data-testid="metric-container"] > label[data-
        testid="stMetricLabel"] > div p {
        font-size: 175% !important;

        </style>
        """,
            unsafe_allow_html=True,
        )
        col1, col2,col3, col4, col5 = st.columns(5,gap='large')

        col1.metric(label='Total Revenue', value=total_revenue, delta=f"
        {percentage_increase}%")

        col2.metric(label='Total Profit
        Margin',value=total_profit_margin,delta= f"
        {readable_percentage_decrease}%")

        col3.metric(label='Total Customers',value=(customers_size).sum())

        col4.metric(label='Best place',value=(best_place))

        col5.metric(label='Best month',value=(best_month))

        st.title('GRAPHS')
        gh1,gh2=st.columns([1,1])

        with gh1:
            st.title('Profit per Year')

             st.line_chart(transactions_by_date_grp_year_total_for_GRAPH[
            'profit_margin'],color='#bc5090',)
        with gh2:
            st.title('Profit per month for last 4 years')
            st.text('This graph tells the sales in each month for last 4
            years')

      st.line_chart(transactions_by_date_grp_avg_month_for_graph[
      'profit_margin'],color='#ffa600')
```

```
        gh3,gh4=st.columns([1,1])

        with gh3:
            st.title('Avg Transactions Per Market')
            st.bar_chart(average_transactions_per_customer_in_market,
            color='#FF9671')

        with gh4:
            st.title('Profit Margin Per Market')
            st.bar_chart(transactions_for_market_grp_total[
            'profit_margin'],color='#845EC2')

            st.write('——')
        pie1,pie2=st.columns([1,1])

        with pie1:
            st.title('Customers count')
            fig1, ax1 = plt.subplots()
            patches, texts, pcts = ax1.pie(customers_count_in_market,labels
            =customers_count_in_market.index, autopct='%1.1f%%',
            shadow=True, startangle=90,pctdistance=0.80)
            fig1.set_facecolor('black')
            ax1.axis('equal')
            for i, patch in enumerate(patches):
                texts[i].set_color(patch.get_facecolor())
            plt.setp(pcts, color='white')
            plt.setp(texts, fontweight=600)
            plt.tight_layout()
            hole = plt.Circle((0, 0), 0.65, facecolor='black')
            plt.gcf().gca().add_artist(hole)
            st.pyplot(fig1)
```

```
        with pie2:
            st.title('Product types share')
            fig2, ax2 = plt.subplots()
            patches1, texts1, pcts1
            =ax2.pie([own_brand_count['product_type']
            ,distribution_count['product_type']],labels=[ '
            OwnBrand','Distribution'], autopct='%1.1f%%',
            shadow=True, startangle=90)
            fig2.set_facecolor('black')
            ax2.axis('equal')
            for i1, patch1 in enumerate(patches1):
                texts1[i1].set_color(patch1.get_facecolor())
            plt.setp(pcts1, color='white')
            plt.setp(texts1, fontweight=600)
            plt.tight_layout()
            st.pyplot(fig2)

        st.write('——')
    else:
        print('oop! we are fail to connect pls try later after sometime')
elif which_file=='N':
    print('You can updata that SQL file with same column, it will work fyn
for whole life')
```

Final look of dashboard

Metric



Line Graph

Bar Graph

Pie Chart