DELHI PUBLIC SCHOOL

Hyderabad

Project Record

AISSCE 2024 Examination

[As a part of the Informatics Practices Course (065)]

SUBMITTED BY:

Name:

[Roll No:]

CERTIFICATE

This is to certify that _____ of class XII Science/Commerce/Humanities, Delhi Public School, Hyderabad, has done this project under my guidance and supervision in the school premises for the academic year 2023-24.
He/ She has shown sincerity and utmost care in the completion of this project. I certify that this project is up to my expectations and as per the guidelines issued by the CBSE.

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

I,_____ do hereby declare that this project is my original work and I would like to thank Ms. Debjani Chatterjee for her wholehearted support and guidance for making it possible to complete this project on time.
I would also like to thank all those who have given me required inputs, my friends and family for their kind support and guidance without which this project would not have been completed.

Name :
Signature:

ABSTRACT

The Warehouse Management System App, offers warehouse managers efficient inventory and order control. It automates tasks, enhances accuracy, and provides insights through charts and reports. Importing/exporting data in CSV format simplifies integration with other systems. Ideal for warehouse managers and those managing inventory and orders.

CONTENTS

System Requirements

Hardware:

Software:

Files Used-
Python - 3.11.4
Visual Studio Code
CSV file - Warehouse_and_Retail_Sales.csv
Github - file sharing

Technology Used

Python :
Python programming language was developed by Guido Van Rossum in February 1991.

Python is an easy-to-learn yet powerful object oriented programming language. It is a very high level programming language yet as powerful as many other middle-level not so high-level languages like C, C++, Java etc.

Pandas :
Pandas is the most popular library in the scientific Python ecosystem for doing data analysis, Pandas is capable of many tasks including: It can read or write in many different data formats (integer, float, double, etc).

It can calculate in all the possible ways data is organized i.e., across rows and down columns.
It can easily select subsets of data from bulky data sets and even combine multiple datasets together. It has functionality to find and fill missing data. It allows you to apply operations to independent groups within the data. It supports reshaping of data into different forms. It supports advanced time-series functionality (Time series forecasting is the use of a model to predict future values based on previously observed values.) It supports visualization by integrating matplotlib and seaborn etc,libraries.

Data Frame :
A Data Frame is a Pandas structure, which stores data in two-dimensional way. It is actually a two-dimensional (tabular and spreadsheet like) labeled array, which is actually an ordered collection of columns where columns may store different types of data, e.g., numeric or string or floating point or Boolean type etc.

Major characteristics of a Data Frame are :
It has two indexes or we can say that two axes - a row index (axis = 0) and a column index (axis = 1).
Conceptually it is like a spreadsheet where each value is identifiable with the combination of row index and column index. The row index is known as index in general and the column index is called the column-name. The indexes can be of numbers or letters or strings. There is no condition of having all data of the same type across columns; its columns can have data of different types. Data Frames are value-mutable and size-mutable.

CSV File :
Refers to the tabular data saved as plaintext where data values are separated by commas.
The CSV format is popular as it offers following advantages : A simple, compact and ubiquitous format for data storage.

A common format for data interchange. It can be opened in popular spreadsheet packages like MS-Excel, Calc etc. and nearly all spreadsheets and databases support import /export to csv format.

Python's Pandas library offers two functions read_csv() and to_csv( ) that help you bring data from a CSV file into a dataframe and write a data frame's data to a CSV file.

Data Visualization and matplotlib :
Data Visualization basically refers to the graphical and visual representation of information and data using visual elements like charts, graphs, and maps, etc. Data Visualization is immensely useful in decision making unveils patterns, trends, outliners, correlations etc. in the data helping decision-makers understand the meaning of data to drive decisions.

The matplotlib is a Python library that provides many interfaces and functionalities for 2D graphics. Matplotlib is a high quality plotting library of python that provides both a very quick way to visualize data from Python and publication-quality figures in many formats. The matplotlib library offers many different named collections of methods; PyPlot is one of such interfaces, a collection of methods within matplotlib which allows users to construct 2D plots easily and interactively.

Tkinter

Tkinter is the standard Python interface to the Tk GUI toolkit.It is the most commonly used method for creating graphical user interfaces with python.

Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit,which is a library for creating desktop applications.With Tkinter, you can create windows,labels,buttons,menus,textboxes, and other widgets.

PandasTables

The pandastable library provides a table widget for Tkinter with plotting and data manipulation functionality. It uses the pandas DataFrame class to store table data. Pandas is an open source Python library providing high-performance data structures and data analysis tools.

CODE & Implementation

General defines and imports

```
import importlib
spec = importlib.util.find_spec("pandastable")
import subprocess
if spec is None:
    subprocess.check_call(["python", "-m", "pip", "install",
"pandastable"])
import tkinter as tk
import sys
from tkinter import ttk,messagebox,font
from pandastable import Table
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

#general defines
dtype_dict = {
    'YEAR': int,
    'MONTH': int,
    'SUPPLIER': str,
    'ITEM CODE': str,
    'ITEM DESCRIPTION': str,
    'ITEM TYPE': str,
    'RETAIL SALES': float,
    'RETAIL TRANSFERS': float,
    'WAREHOUSE SALES': float}
```

Creating Function for the login page

```
  def create_login_window(root):
      window = tk.Toplevel(root)
      window.attributes('-topmost', True)
      window.lift()
      style = ttk.Style()
      screen_width = window.winfo_screenwidth()
      screen_height = window.winfo_screenheight()
      window.title("Login Screen")
      window.resizable(True, True)
      window_height = 800
      window_width = 800
      screen_width = window.winfo_screenwidth()
      screen_height = window.winfo_screenheight()
      position_top = int(screen_height // 2 - window_height // 2)
      position_right = int(screen_width // 2 - window_width // 2)
      window.geometry(f"{window_width}x{window_height}+{position_right}+
  {position_top}")

      def resize(event):
```

```
        # Calculate new font size based on window size
        new_size = max(8, min(int(event.width / 50), int(event.height
/ 30)))
        style.configure("TButton", font=("Rockwell", new_size))
        style.configure("TEntry", font=("Rockwell", new_size))

    window.bind('<Configure>', resize)
    # background color
    window.configure(bg='#61A0AF')

    # style creation
    style = ttk.Style()
    style.configure("TEntry",
                    foreground="#F06C9B",
                    fieldbackground="#F06C9B",
                    bordercolor="#F06C9B",
                    lightcolor="#F06C9B",
                    darkcolor="#F06C9B",
                    borderwidth=20,
                    relief="groove",
                    font=("Rockwell", 14))

    style.configure("TButton",
                    background="#F5D491",
                    foreground="black",
                    borderwidth=20,
                    relief="groove",
                    font=("Rockwell", 14))



    username_label = ttk.Label(window, text="Username",
font=("Rockwell", 14), background='#61A0AF',
foreground="blue",anchor='center', justify='center')
    username_label.pack(fill=tk.BOTH, expand=1)
    username_entry = ttk.Entry(window,style='TEntry')
    username_entry.pack(fill=tk.BOTH, expand=1)

    password_label = ttk.Label(window,
text="Password",font=("Rockwell", 14), background='#61A0AF',
foreground="blue",anchor='center', justify='center')
    password_label.pack(fill=tk.BOTH, expand=1)
    password_entry = ttk.Entry(window,style="TEntry",show="*")
    password_entry.pack(fill=tk.BOTH, expand=1)

    # Adding validation logic here
    def validate_login(username, password):
        if username == "admin" and password == "123":
                window.attributes('-topmost', False)
                messagebox.showinfo("Login info", "Welcome Admin!")
```

```
                window.destroy()
                create_data_window(root)
        else:
            window.attributes('-topmost', False)
            messagebox.showinfo("Login info", "Incorrect credentials")
            window.destroy()
            sys.exit()

    submit_button = ttk.Button(window, text="Login",style="TButton",
command=lambda: validate_login(username_entry.get(),
password_entry.get()))
    submit_button.pack(fill=tk.BOTH, expand=1)



    def on_close():
        window.destroy()
        sys.exit()
    window.protocol('WM_DELETE_WINDOW',on_close)
```

Creating Loading Page function

```
#Creating loading window

def create_loading_window(root):
    loading_window = tk.Toplevel(root)
    loading_window.lift()
    loading_window.title("Loading...")
    screen_width = loading_window.winfo_screenwidth()
    screen_height = loading_window.winfo_screenheight()
    window_height = 300
    window_width = 300
    loading_window.resizable(False, False)
    position_top = int(screen_height // 2 - window_height // 2)
    position_right = int(screen_width // 2 - window_width // 2)
    loading_window.geometry(f"{window_width}x{window_height}+
{position_right}+{position_top}")

    # Add a label with a custom font and color
    loading_label = ttk.Label(loading_window, text="Loading data,
please wait...", font=("Helvetica", 16), foreground="blue")
    loading_label.pack(pady=10)

    # Add a progress bar
    progress = ttk.Progressbar(loading_window, length=200,
mode='determinate')
    progress.pack(pady=10)

    return loading_window, progress
```

```
# loading data in chunks and reporting progress

def load_data(filename, loading_window, progress):
    chunksize = 10000
    chunks = []
    total_rows = sum(1 for row in open(filename, 'r')) - 1 # no of
rows - header

    for i, chunk in enumerate(pd.read_csv(filename,
chunksize=chunksize)):
        chunks.append(chunk)

        #update progress bar
        progress['value'] = (i+1) * chunksize / total_rows * 100
        loading_window.update()
    # concatenate all chunks into one dataframe
    data = pd.concat(chunks, axis=0)
    data = data.dropna()
    data['DATE']= pd.to_datetime(data[['YEAR','MONTH']].assign(DAY=1))
    data = data.sort_values(by='DATE')
    # Group the data by 'DATE' and sum the 'RETAIL SALES' for each
date
    sales_data = data.groupby('DATE')['RETAIL
SALES'].sum().reset_index()

    # delete loading window(doing it here to make the switch seem
instantaneous)
    loading_window.destroy()

    return(data,sales_data)

def treeview_sort_column(tv, col, reverse):
    l = [(tv.set(k, col), k) for k in tv.get_children('')]
    l.sort(reverse=reverse)

    #rearrange items in sorted position
    for index, (val, k) in enumerate(l):
        tv.move(k,'', index)
    # reverse sort next time
    tv.heading(col, command=lambda: treeview_sort_column(tv, col, not
reverse))
#paging defs
items_per_page = 10000
current_page = 0

def display_page(page):
    # Clear the treeview
    for i in tree.get_children():
        tree.delete(i)
```

```
    # Calculate the range of items to display
    start = page * items_per_page
    end = start + items_per_page

    # Insert the items for this page into the treeview
    for item in items[start:end]:
        tree.insert('', 'end', values=item)

def next_page():
    global current_page
    current_page += 1
    display_page(current_page)

def prev_page():
    global current_page
    if current_page > 0:
        current_page -= 1
        display_page(current_page)
```

Creating Progress bar function

```
def create_data_window(root):
    #create progress bar
    loading_window, progress = create_loading_window(root)
    # Load the data
    data,sales_data = load_data('Warehouse_and_Retail_Sales.csv',
loading_window, progress)

    # Group the data by year and sum the 'RETAIL SALES' for each year
    yearly_sales = data.groupby(data['DATE'].dt.year)['RETAIL
SALES'].sum()
```

Creating Function for main window

```
    # Create a new tkinter window
    global window
    window = tk.Toplevel(root)
    window.lift()
    window.title("Data Window")
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()
    window_width = int(screen_width * 1)
    window_height = int(screen_height * 0.8)
    position_top = int(screen_height // 2 - window_height // 2)
    position_right = int(screen_width // 2 - window_width // 2)
    window.geometry(f"{window_width}x{window_height}+{position_right}+
{position_top}")
    # Create a Frame for the Treeview
    tree_frame = tk.Frame(window)
    tree_frame.grid(row=0,column=0,pady = 10,sticky='n')
    tree_frame.grid_propagate(True)
```

Creating function for scrollbar

```
    # create a scrollbar
    scrollbar = ttk.Scrollbar(tree_frame,)
    scrollbar.grid(row=0,column=1,sticky='ns')
    #columns to Display
    column_dis = [col for col in data.columns if col != 'DATE']
    #Create a Treeview widget and display the data in it
    global tree
    tree = ttk.Treeview(tree_frame, columns=column_dis, show =
'headings', yscrollcommand=scrollbar.set)
    for column in column_dis:
            tree.heading(column, text=column, command=lambda
_col=column: treeview_sort_column(tree, _col, False))
    # Create a list of items
    global items
    items = [tuple(x) for x in data.values]

    # Display the first page of items
    display_page(current_page)

    tree.grid(row=0, column=0, sticky='nsew')
    scrollbar.config(command=tree.yview)
    # Add buttons to go to the next and previous pages
    prev_button = ttk.Button(window, text="Previous Page",
command=prev_page)
    prev_button.grid(row=0, column=0, sticky='sw', pady= 5)

    next_button = ttk.Button(window, text="Next Page",
command=next_page)
    next_button.grid(row=0, column=0, sticky='se', pady= 5)
```

Creating the bar plot subchart

```
    # Create a Frame for the chart
    chart_frame = tk.Frame(window,width=1000,height=200)
    chart_frame.grid(row=1, column=0)
    chart_frame.grid_propagate(True)
    # Format 'DATE' to include both month and year
    sales_data['DATE'] = sales_data['DATE'].dt.strftime('%Y-%m')
    # Create a chart and display the sales data in it
    fig = plt.Figure(dpi=100)
    ax = fig.add_subplot(111)
    bars = ax.bar(sales_data['DATE'], round(sales_data['RETAIL
SALES']),width=0.75, color='skyblue', edgecolor='grey', zorder=3)
    ax.bar_label(bars, rotation=90)
    ax.set_xlabel('Year,Month')
    ax.set_ylabel('Retail Sales')
    ax.set_ylim([0,round(max(sales_data['RETAIL SALES']))+30000])


    # Rotate x-axis labels
    labels = ax.get_xticklabels()
```

```
    plt.setp(labels, rotation=45)
    fig.tight_layout()
    #Draw the chart on the window
    chart = FigureCanvasTkAgg(fig, master=chart_frame)
    chart.draw()
    chart.get_tk_widget().grid(row=0,column=0, sticky ='nsew')
```

Creating the detailed analysis

```
    # Create a Frame for the detailed analysis
    analysis_frame = tk.Frame(window, bd=2, relief='groove')
    analysis_frame.grid(row=1,column=0,sticky='w')
    analysis_frame.grid_propagate(True)

    # Create a Text widget and display the detailed analysis in it
    analysis = tk.Text(analysis_frame, height=30, width=50)
    analysis.grid(row=0,column=0, sticky ='nsew')

     # Configure a tag for bold text
    analysis.tag_configure('bold', font=('Rockwell', 20, 'bold'))

    analysis.insert(tk.END, "Here is the detailed analysis:\n\n",
'bold')
    analysis.insert(tk.END, "Total sales: " + str(round(data['RETAIL
SALES'].sum())) + "\n")
    analysis.insert(tk.END, "Average monthly sales for all time: " +
str(round(data['RETAIL SALES'].mean())) + "\n")

    #getting list of years
    sales_data['DATE'] = pd.to_datetime(sales_data['DATE'])
    years = sorted(sales_data['DATE'].dt.year.unique())

    # Create a StringVar to hold the selected year
    selected_year = tk.StringVar(window)
    selected_year.set(years[0])  # default value

    # Create an OptionMenu for the years
    year_menu = tk.OptionMenu(window, selected_year, *years)
    year_menu.grid(row=1, column=0,sticky='ws',padx=10)

        # Function to calculate and display average monthly sales for
a specific year
    def calculate_average_monthly_sales():
        analysis.delete('1.0',tk.END)
        analysis.insert(tk.END, "Here is the detailed analysis:\n\n")
        analysis.insert(tk.END, "Total sales: " +
str(round(data['RETAIL SALES'].sum())) + "\n")
        analysis.insert(tk.END, "Average monthly sales for all time: "
+ str(round(data['RETAIL SALES'].mean())) + "\n")
```

```
        year = selected_year.get()
        sales_data_year = sales_data[sales_data['DATE'].dt.year ==
int(year)]
        average_monthly_sales =
sales_data_year.groupby(sales_data_year['DATE'].dt.month)['RETAIL
SALES'].mean()
        month_names = {1: 'January', 2: 'February', 3: 'March', 4:
'April', 5: 'May', 6: 'June',
                       7: 'July', 8: 'August', 9: 'September', 10:
'October', 11: 'November', 12: 'December'}
        average_monthly_sales.index =
average_monthly_sales.index.map(month_names)
        average_monthly_sales_str = str(average_monthly_sales)
        average_monthly_sales_str = '\
n'.join(average_monthly_sales_str.split('\n')[:-2])
        analysis.insert(tk.END, '\n\nAverage Monthly Sales for ' +
year + ':\n' + average_monthly_sales_str)


    # Add a button to calculate average monthly sales
    calculate_button = tk.Button(window, text="Calculate Average
Monthly Sales", command=calculate_average_monthly_sales)
    calculate_button.grid(row=2, column=0, columnspan=2, sticky='wn')
```

Creating a function for additional analysis

```
    # Additional detailed analysis
    # Create a Frame for the additional detailed analysis
    add_analysis_frame = tk.Frame(window, bd=2, relief='groove')
    add_analysis_frame.grid(row=1,column=0,sticky='nes')
    add_analysis_frame.grid_propagate(True)

    # Create a Text widget and display the additional detailed
analysis in it
    add_analysis= tk.Text(add_analysis_frame,height=30, width=50)
    add_analysis.grid(row=0,column=0, sticky ='nsew')
    add_analysis.tag_configure('bold', font=('Rockwell', 20, 'bold'))
    add_analysis.insert(tk.END, 'Additional Details:\n','bold')

    # Calculate and display sales growth
    sales_growth = sales_data['RETAIL SALES'].pct_change().mean()
    add_analysis.insert(tk.END, '\n Percentage Sales Growth: ' +
str(100*sales_growth)+':\n')


    # Calculate and display average deal size
    average_deal_size = round(sales_data['RETAIL SALES'].mean())
    add_analysis.insert(tk.END, '\nAverage Deal Size: ' +
str(average_deal_size)+':\n')
```

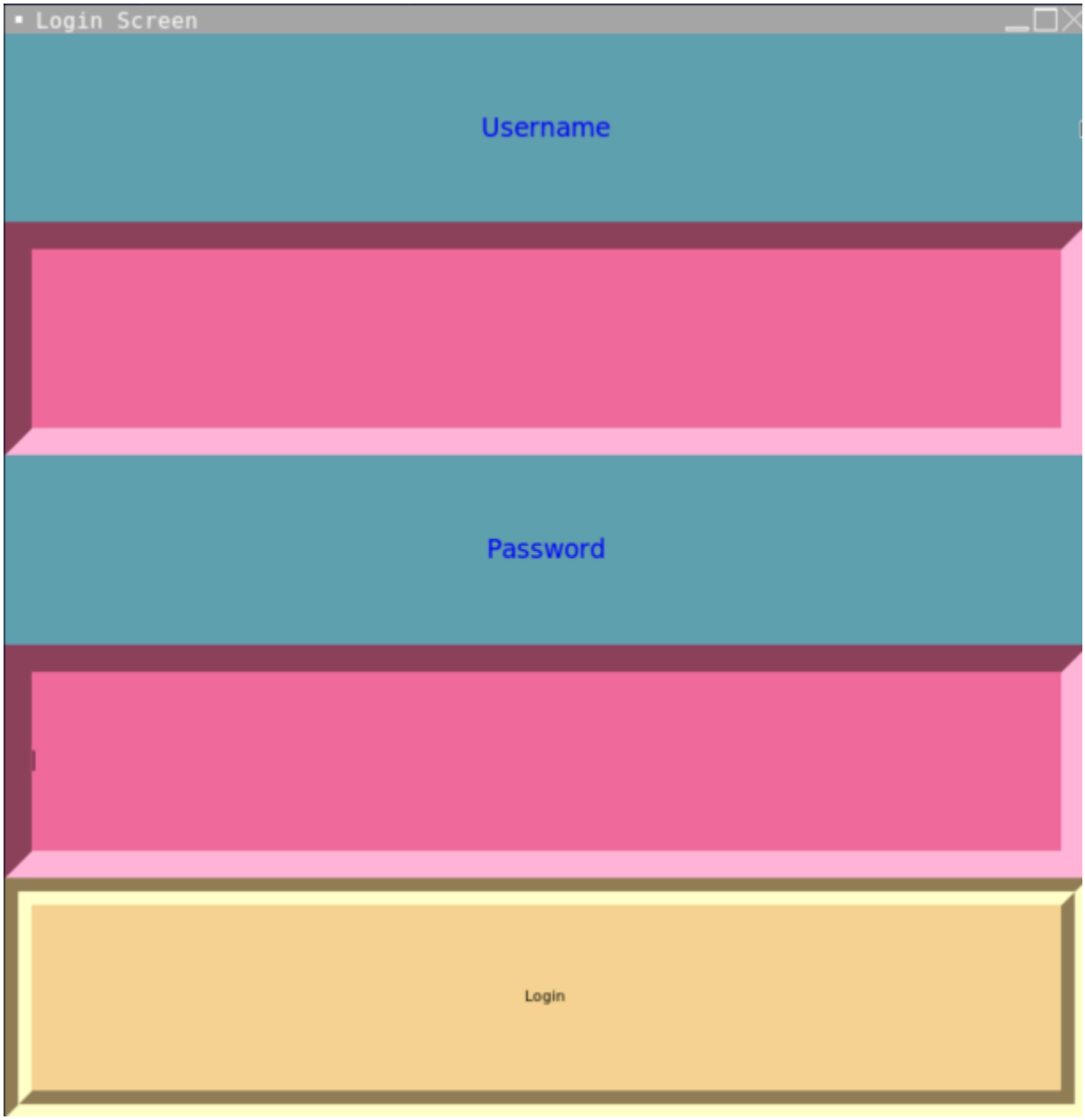
```
    def on_close():
        window.destroy()
        sys.exit()
    window.protocol('WM_DELETE_WINDOW',on_close)


def main():
    root = tk.Tk()
    root.withdraw()
    root.lift()
    create_login_window(root)
    tk.mainloop()



if __name__ == "__main__":
        main()
```

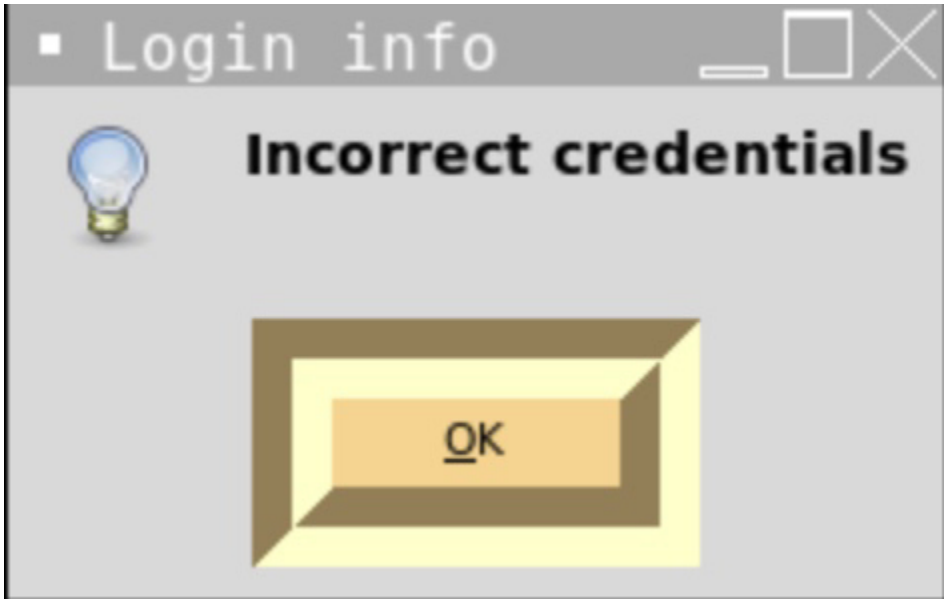
Login page



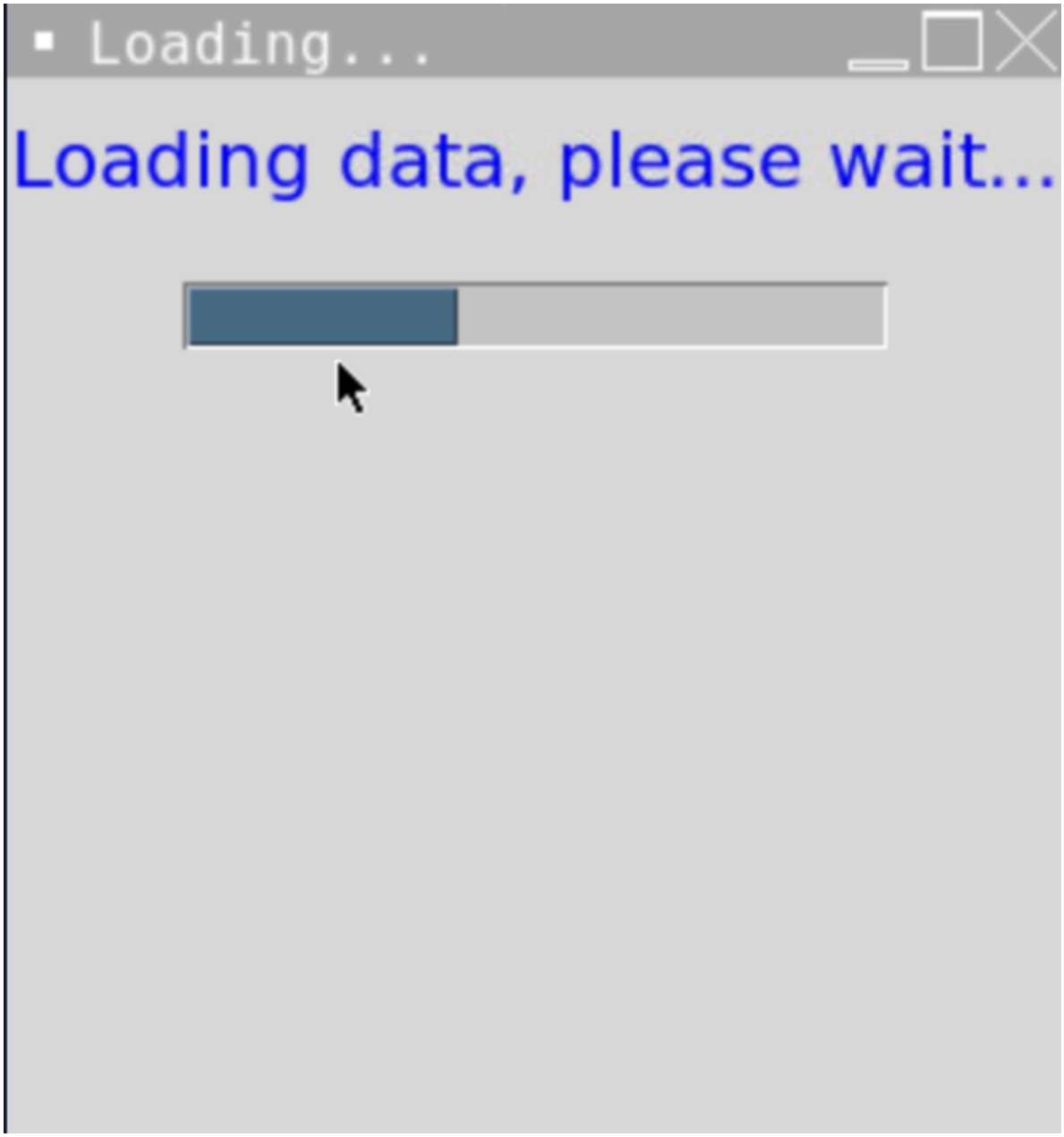
Login validation

Correct login

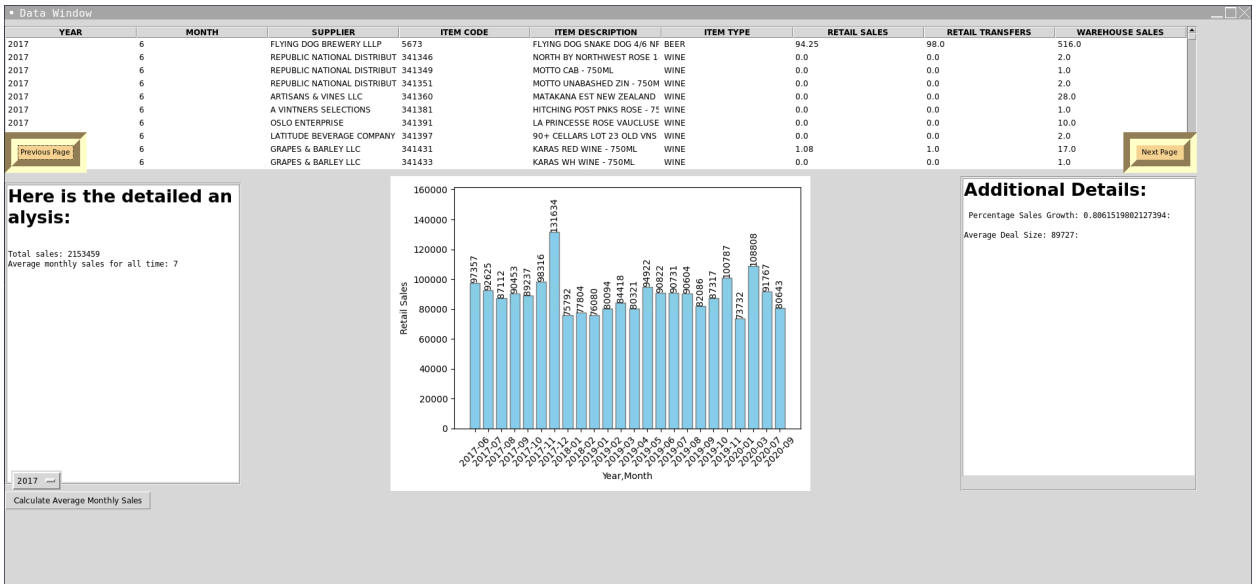
Incorrect login



Loading Page



Main Page


Data Window
YEAR
MONTH
SUPPLIER
ITEM CODE
ITEM DESCRIPTION
ITEM TYPE
RETAIL SALES
RETAIL TRANSFERS
WAREHOUSE SALES
2017 6 FLYING DOG BREWERY LLLP 5673 FLYING DOG SNAKE DOG 4/6 NF BEER 94.25 98.0 516.0
2017 6 REPUBLIC NATIONAL DISTRIBUT 341346 NORTH BY NORTHWEST ROSE 1 WINE 0.0 0.0 2.0
2017 6 REPUBLIC NATIONAL DISTRIBUT 341349 MOTTO CAB - 750ML WINE 0.0 0.0 1.0
2017 6 REPUBLIC NATIONAL DISTRIBUT 341351 MOTTO UNABASHED ZIN - 750M WINE 0.0 0.0 2.0
2017 6 ARTISANS & VINES LLC 341360 MATAKANA EST NEW ZEALAND WINE 0.0 0.0 28.0
2017 6 A VINTNERS SELECTIONS 341381 HITCHING POST PNKS ROSE - 75 WINE 0.0 0.0 1.0
2017 6 OSLO ENTERPRISE 341391 LA PRINCESSE ROSE VAUCLUSE WINE 0.0 0.0 10.0
6 LATITUDE BEVERAGE COMPANY 341397 90+ CELLARS LOT 23 OLD VNS WINE 0.0 0.0 2.0
6 GRAPES & BARLEY LLC 341431 KARAS RED WINE - 750ML WINE 1.08 1.0 17.0
6 GRAPES & BARLEY LLC 341433 KARAS WH WINE - 750ML WINE 0.0 0.0 1.0
Previous Page
Next Page
Here is the detailed an alysis:
Total sales: 2153459
Average monthly sales for all time: 7
2017
Calculate Average Monthly Sales
Retail Sales
Year,Month
Additional Details:
Percentage Sales Growth: 0.8061519802127394:
Average Deal Size: 89727:

Main - Data table

Page-1

Data Window

YEAR	MONTH	SUPPLIER	ITEM CODE	ITEM DESCRIPTION	ITEM TYPE	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
2017	6	FLYING DOG BREWERY LLLP	5673	FLYING DOG SNAKE DOG 4/6 NF	BEER	94.25	98.0	516.0
2017	6	REPUBLIC NATIONAL DISTRIBUT	341346	NORTH BY NORTHWEST ROSE 1	WINE	0.0	0.0	2.0
2017	6	REPUBLIC NATIONAL DISTRIBUT	341349	MOTTO CAB - 750ML	WINE	0.0	0.0	1.0
2017	6	REPUBLIC NATIONAL DISTRIBUT	341351	MOTTO UNABASHED ZIN - 750M	WINE	0.0	0.0	2.0
2017	6	ARTISANS & VINES LLC	341360	MATAKANA EST NEW ZEALAND	WINE	0.0	0.0	28.0
2017	6	A VINTNERS SELECTIONS	341381	HITCHING POST PNKS ROSE - 75	WINE	0.0	0.0	1.0
2017	6	OSLO ENTERPRISE	341391	LA PRINCESSE ROSE VAUCLUSE	WINE	0.0	0.0	10.0
	6	LATITUDE BEVERAGE COMPANY	341397	90+ CELLARS LOT 23 OLD VNS	WINE	0.0	0.0	2.0
Previous Page	6	GRAPES & BARLEY LLC	341431	KARAS RED WINE - 750ML	WINE	1.08	1.0	17.0
	6	GRAPES & BARLEY LLC	341433	KARAS WH WINE - 750ML	WINE	0.0	0.0	1.0

Next Page

Page-2

Data Window

YEAR	MONTH	SUPPLIER	ITEM CODE	ITEM DESCRIPTION	ITEM TYPE	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
2017	6	KLEIN FAMILY VINTNERS	82295	R STRONG CHALK HILL CHARD -	WINE	12.59	13.0	3.0
2017	6	E & J GALLO WINERY	82301	GALLO FAMILY VYDS SWEET BE	WINE	3.47	6.0	9.0
2017	6	REPUBLIC NATIONAL DISTRIBUT	82303	ZACO TEMP - 750ML	WINE	0.65	0.0	0.0
2017	6	DEUTSCH FAMILY WINE & SPIRI	82313	SKYFALL COLUMBIA VLY CAB - 7	WINE	1.22	0.0	3.0
2017	6	A VINTNERS SELECTIONS	82314	ROOT 1 CARMENERE - 750ML	WINE	6.55	4.0	3.0
2017	6	AMERICAN BEVERAGE CORPORA	82329	DAILYS MAUI WOWIE - 10.OZ	BEER	8.31	5.96	4.0
2017	6	E & J GALLO WINERY	82333	CARLO ROSSI CHIANTI - 4L	WINE	17.75	18.75	22.0
	6	A VINTNERS SELECTIONS	82348	HIGH NOTE MALBEC - 750ML	WINE	3.81	5.0	6.0
Previous Page	6	FRANCIS COPPOLA WINERY LLC	82353	THE WHITE DOE - 750ML	WINE	1.82	0.0	0.0
	6	DEUTSCH FAMILY WINE & SPIRI	82355	SKYFALL COLUMBIA VLY RED BL	WINE	0.48	0.0	1.0

Next Page

Page-3

YEAR	MONTH	SUPPLIER	ITEM CODE	ITEM DESCRIPTION	ITEM TYPE	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
2017	7	DIONYSOS IMPORTS INC	77470	GATAO VINHO VERDE - 750ML	WINE	18.11	16.0	12.0
2017	7	REPUBLIC NATIONAL DISTRIBUT	77488	TAYLOR LAKE COUNTRY WH - 3L	WINE	1.75	0.0	0.0
2017	7	PRESTIGE BEVERAGE GROUP OF	77501	ASOMBROSO TEQUILA - SILVER	LIQUOR	0.17	0.0	0.0
2017	7	CAMPARI AMERICA LLC	77280	GRAND MARNIER 1880 - 750ML	LIQUOR	0.34	0.0	0.0
2017	7	SAZERAC CO	77273	PADDYS IRISH WHISKEY - 750M	LIQUOR	0.33	0.0	0.0
2017	7	JIM BEAM BRANDS CO	77256	JIM BEAM HONEY - 750ML	LIQUOR	38.01	35.0	0.0
2017	7	PERNOD RICARD USA LLC	77241	ABSOLUT VODKA - HIBISKUS - 7	LIQUOR	0.33	0.0	0.0
	7	DOPS INC	76830	HORTON VIOG - 750ML	WINE	0.81	0.0	0.0
Previous Page	7	REPUBLIC NATIONAL DISTRIBUT	76857	VAN GOGH GIN - 750ML	LIQUOR	0.0	2.0	0.0
	7	REPUBLIC NATIONAL DISTRIBUT	76868	FINEST CALL PEACH PUREE - 1L	NON-ALCOHOL	2.52	3.0	0.0

Next Page

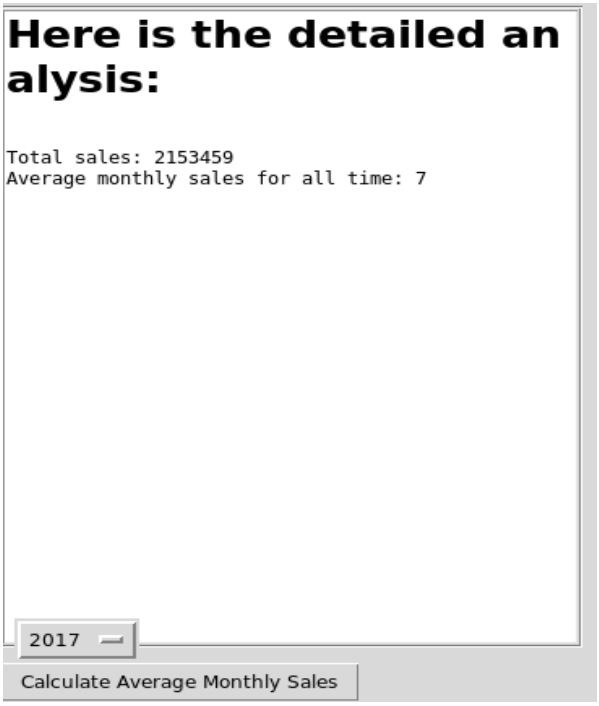
Page-4

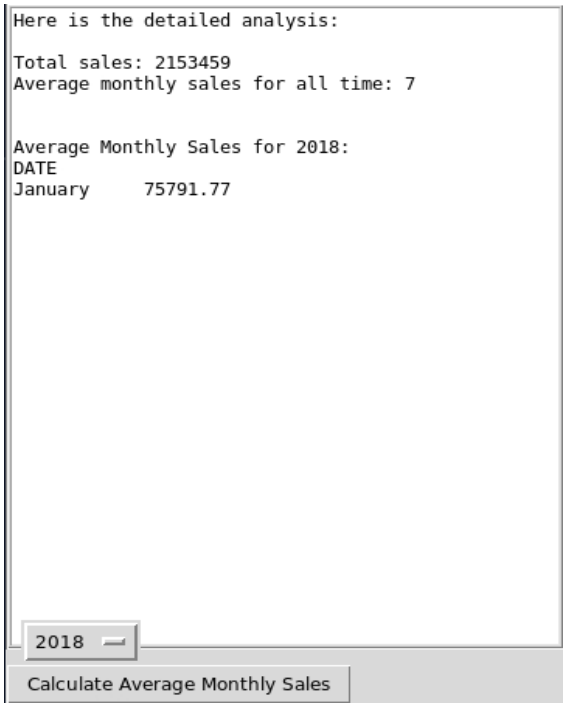
Data Window

YEAR	MONTH	SUPPLIER	ITEM CODE	ITEM DESCRIPTION	ITEM TYPE	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
2017	8	A VINTNERS SELECTIONS	305254	BADENHORST SECATEURS RED	WINE	0.32	0.0	0.0
2017	8	TRI-VIN IMPORTS	305240	DREAMFISH S/BLC - 750ML	WINE	0.0	0.0	5.0
2017	8	TRI-VIN IMPORTS	305209	DREAMBIRD P/NOIR - 750ML	WINE	0.58	0.0	0.0
2017	8	LUXCO SPIRITED BRANDS	30520	CALVERT GIN - 1.75L	LIQUOR	1.52	1.0	0.0
2017	8	KYSELA PERE ET FILS LTD	305195	DOM LA SALETTE GASCOGNE - 7	WINE	0.0	0.0	5.0
2017	8	MONSIEUR TOUTON SELECTION	305149	LE PETIT ROUVIERE RSE - 750M	WINE	0.0	0.0	2.0
2017	8	ELITE WINES IMPORTS	305146	DOM BELLEVUE ROSE - 750ML	WINE	1.56	3.0	26.0
	8	LUNEAU USA INC	305127	LOUIS LAURENT VOUV - 750ML	WINE	0.16	0.0	0.0
Previous Page	8	ELITE WINES IMPORTS	305120	DOM BERTHET RAYNE CDP RGE	WINE	0.41	1.0	2.0
	8	REPUBLIC NATIONAL DISTRIBUT	305112	HOGUE S/BLC - 750ML	WINE	0.0	0.0	3.0

Next Page

Detailed analysis of monthly sales





Additional Details

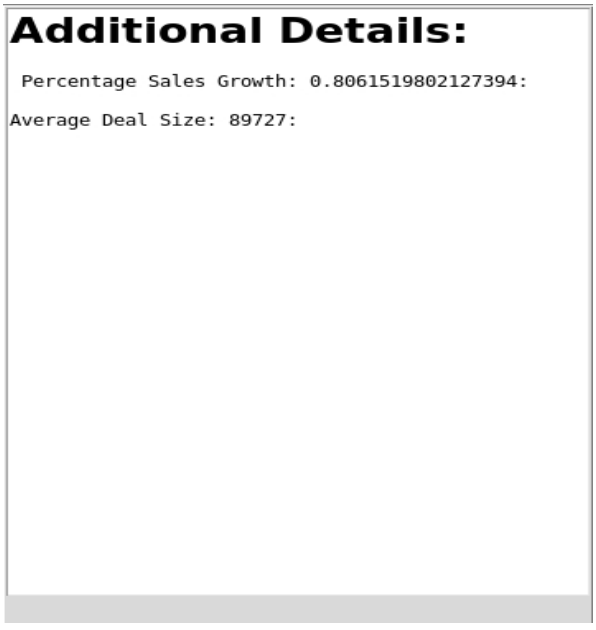
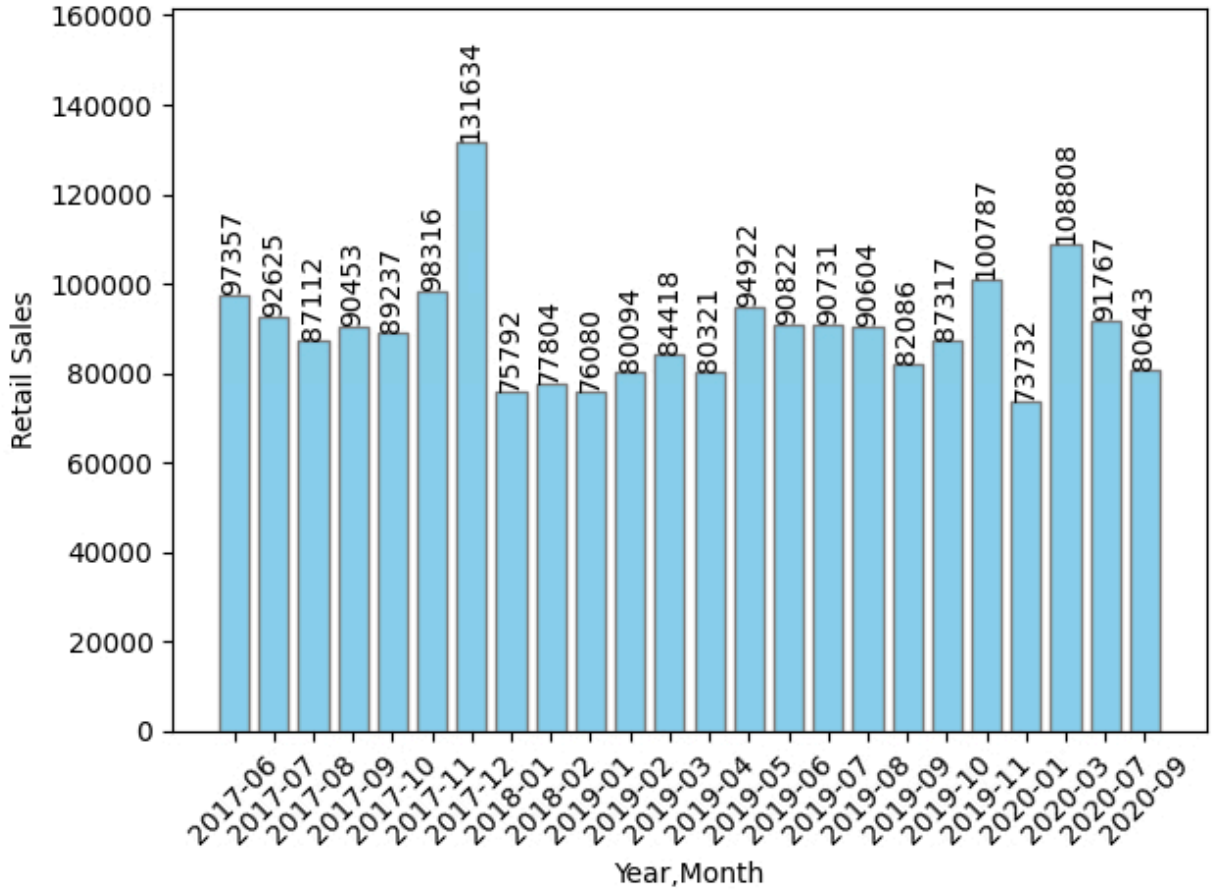


Chart for Sales vs Months of Year



Conclusion

The Warehouse Management System App has the potential to revolutionize warehouse management by offering efficient inventory and order control, automated tasks, enhanced accuracy, and insightful data visualization. This combination of features can significantly improve warehouse efficiency, reduce costs, and optimize operations.

By automating repetitive tasks and providing real-time visibility into inventory levels and order status, the app empowers warehouse managers to make informed decisions and optimize resource allocation. Additionally, the seamless integration with other systems through CSV import/export functionality further enhances the app's value and flexibility.

For warehouse managers and those managing inventory and orders, the Warehouse Management System App presents an invaluable tool for streamlining operations, eliminating errors, and gaining valuable insights to drive business growth

Bibliography

Books and software used:-

- Informatics Practices Class XI and Class XII by Sumita Arora.
- https://www.pexels.com/photo/javascript-code-1972464/
- Visual Studio Code
- https://www.geeksforgeeks.org/introduction-to-tkinter/
- https://www.geeksforgeeks.org/pandas-tutorial/
- https://catalog.data.gov/dataset/warehouse-and-retail-sales