

# SHANTI ASIATIC SCHOOL

BEHIND RAF CAMP, NR. BALIYADEV TEMPLE, VASTRAL, AHMEDABAD, GUJARAT - 382418

**ACADEMIC YEAR – 2023-2024**



## **A PROJECT REPORT ON Mini Accounting & Inventory Management**

### **Submitted by:-**

Name : **Jainil Jayswal**  
Roll No :  
Class : XII  
Subject : Computer Science  
Subject Code : 083

### **Submitted to:-**

Ms. Pooja Nirmal  
PGT – Computer Science





# SHANTI ASIATIC SCHOOL, VASTRAL

BEHIND RAF CAMP, NR. BALIYADEV TEMPLE, VASTRAL,  
AHMEDABAD, GUJARAT - 382418

## CERTIFICATE

This is to certify that Jainil Jayswal  
**Roll No:** \_\_\_\_\_ has successfully completed the  
project work entitled **Mini Accounting & Inventory**  
**Management** in the subject Computer Science (083) under  
the supervision of **Ms. Pooja Nirmal** laid down in the  
regulations of CBSE for the purpose of Practical  
Examination in Class XII to be held in **Shanti Asiatic**  
**School, Vastral, Ahmedabad.**

Internal Examiner:

Name: \_\_\_\_\_

Signature:

External Examiner:

Name: \_\_\_\_\_

Signature:

Principal:

Name: \_\_\_\_\_

Signature:

# ACKNOWLEDGEMENT

I would like to express a deep sense of thanks & gratitude to my teacher **Ms. Pooja Nirmal** for guiding me immensely through the course of the project. She always evidences keen interest in my work. Her constructive advice and constant motivation have been responsible for the successful completion of this project.

I express my sincere thanks to **Ms. Suchitra Vijayraghavan**, Principal of Shanti Asiatic School, Vastral for her co-ordination in extending every possible support for the completion of this project.

I would also like to thank to my parents for their motivation and support. I must thank to my classmate for their timely help and support for this project.

**Jainil Jayswal**

Class: XII

# TABLE OF CONTENTS

Sr.No.	CONTENTS	Page No.
1.	Introduction	5
2.	Objectives and Scope of the Project	6
3.	Hardware Requirements	7
4.	Software Requirements	7
5.	MySQL Database Connection	8
6.	MySQL Table	9
7.	Features	13
8.	Source Code	14
9.	Output Screen	39
10.	Future Scope	47
11.	Bibliography	47

# INTRODUCTION

In the dynamic landscape of modern business, where agility and precision are paramount, the integration of advanced technological solutions becomes not just a choice but a necessity. This project marks a significant stride in meeting this demand, presenting the **Mini Accounting & Inventory Management Software**. Developed in Python with MySQL integration, this software emerges as a versatile tool, addressing the intricate needs of businesses in managing their financial and inventory affairs efficiently.

## Background

The roots of this endeavour trace back to the challenges faced by businesses—particularly small and medium-sized enterprises (SMEs)—in balancing the intricacies of accounting and inventory management. Conventional methods often prove time-consuming, error-prone, and lack the agility required to adapt to the fast-paced business environment. Recognizing this gap, our project sets out to bridge it with an innovative and comprehensive software solution.

# **Objective & Scope of the Project**

## **Objectives:**

1. Develop a user-friendly accounting and inventory management system.
2. Enable seamless tracking and management of financial transactions.
3. Provide real-time visibility into inventory levels and transactions.
4. Streamline reporting for informed decision-making.

## **Scope**

The project encompasses the development of a desktop-based application, offering a centralized platform for accounting and inventory management. Key features include transaction recording, financial reporting, inventory tracking, and user authentication.

## **Hardware Requirements**

- A computer/laptop with Operating System-Windows 7 or above
- X86 64-bit CPU-Intel/AMD Architecture
- **Processor:** Dual-core processor or higher
- **RAM:** 4 GB or higher
- **Storage:** 5 GB of free disk space

## **Software Requirements**

- Python 3.8.x or Higher Version
- MySQL
- MySQL Connector Python

# MySQL Database Connection

Create a database in MySQL named “AI”

```
mysql> show databases
-> ;
+-----+
| Database |
+-----+
| ai       |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.00 sec)

mysql> use AI;
Database changed
mysql> show tables
-> ;
+-----+
| Tables_in_ai |
+-----+
| cashbook     |
| itemname     |
| partyname    |
| payment      |
| production   |
| purchase     |
| receipt      |
| sales        |
+-----+
8 rows in set (0.00 sec)

mysql>
```



# MySQL Table

## Table 1 - Cash Book

```
mysql> desc cashbook;
```

Field	Type	Null	Key	Default	Extra
DrDate	date	YES		NULL	
DrParticulars	varchar(30)	YES		NULL	
DrAmount	int	YES		NULL	
CrDate	date	YES		NULL	
CrParticulars	varchar(30)	YES		NULL	
CrAmount	int	YES		NULL	

```
6 rows in set (0.01 sec)
```

## Table 2 - Item Name

```
mysql> desc itemname;
```

Field	Type	Null	Key	Default	Extra
Item	varchar(30)	NO	PRI	NULL	
Unit	varchar(6)	NO		NULL	

```
2 rows in set (0.00 sec)
```

**Table 3 - Party Name**

```
mysql> desc partyname;
```

Field	Type	Null	Key	Default	Extra
Party	varchar(30)	NO	PRI	NULL	
Address	varchar(50)	NO		NULL	
MobileNo	int	NO		NULL	

3 rows in set (0.00 sec)

**Table 4 – Payment**

```
mysql> desc payment;
```

Field	Type	Null	Key	Default	Extra
Date	date	NO		NULL	
VCH_NO	int	NO		NULL	
Party_Name	varchar(30)	NO		NULL	
Amount	int	NO		NULL	

4 rows in set (0.00 sec)

## Table 5 – Production

```
mysql> desc production;
```

Field	Type	Null	Key	Default	Extra
date	date	YES		NULL	
Item_consumed	varchar(30)	YES		NULL	
Qty_consumed	int	YES		NULL	
Unit_consumed	varchar(10)	YES		NULL	
Item_generated	varchar(30)	YES		NULL	
Qty_generated	int	YES		NULL	
Unit_generated	varchar(10)	YES		NULL	

```
7 rows in set (0.00 sec)
```

## Table 6 – Purchase

```
mysql> desc purchase;
```

Field	Type	Null	Key	Default	Extra
Date	date	NO		NULL	
VCH_NO	int	NO		NULL	
Party_Name	varchar(30)	NO		NULL	
Item_Name	varchar(30)	NO		NULL	
Quantity	int	NO		NULL	
Unit	varchar(10)	NO		NULL	
Price	int	NO		NULL	
Amount	int	NO		NULL	

```
8 rows in set (0.00 sec)
```

## Table 7 – Receipt

```
mysql> desc receipt;
```

Field	Type	Null	Key	Default	Extra
Date	date	NO		NULL	
VCH_NO	int	NO		NULL	
Party_Name	varchar(30)	NO		NULL	
Amount	int	NO		NULL	

```
4 rows in set (0.00 sec)
```

## Table 8 – Sales

```
mysql> desc sales;
```

Field	Type	Null	Key	Default	Extra
Date	date	NO		NULL	
VCH_NO	int	NO		NULL	
Party_Name	varchar(30)	NO		NULL	
Item_Name	varchar(30)	NO		NULL	
Quantity	int	NO		NULL	
Unit	varchar(10)	NO		NULL	
Price	int	NO		NULL	
Amount	int	NO		NULL	

```
8 rows in set (0.00 sec)
```

# **Features**

## **1. Accounting Module:**

- . Record financial transactions.
- . Generate financial reports.

## **2. Inventory Management:**

- . Track product inventory in real-time.
- . Manage stock levels.

## **3. User Authentication:**

- . Secure access with user authentication.

## **4. Graphical User Interface (GUI):**

- . Intuitive and user-friendly design.

# Source Code

AI.py

```
print("_____Toh Kaise Hai APP Log!!!_____")
j='continue'
from accounting import accounting_call
from inventory import inventory_call
from masters import masters_call
print("_____Mini Accounting & Inventory Management Software_____")
while j=='continue':
    print("  1.Accounting          2.Inventory          3.Masters          4.Exit")
    ch=int(input("Choose An Option : "))
    if ch==1:
        accounting_call()
    elif ch==2:
        inventory_call()
    elif ch==3:
        masters_call()
    elif ch==4:
        j='.'
    else:
        print("Please Enter Valid Choice!")
```

# accounting.py

```
def accounting_call():
    from sales import sales_add, sales_list
    from purchase import purchase_add, purchase_list
    from payment import payment_add, payment_list
    from receipt import receipt_add, receipt_list
    from ledger import ledger
    from cash_book import cash_book
    while True:
        print("1.Sales  2.Purchase  3.Payment  4.Receipt  5.Ledger  6.Cash Book  7.Back")
        ich = int(input("Enter Your choice : "))
        if ich == 1:
            while True:
                print("1.Add", "2.List", "3.Back", sep='\t')
                iich = int(input("Enter Your Choice : "))
                if iich == 1:
                    sales_add()
                elif iich == 2:
                    sales_list()
                elif iich == 3:
                    break
                else:
                    print("Enter Valid Choice!!")
            elif ich == 2:
                while True:
                    print("1.Add", "2.List", "3.Back", sep='\t')
                    iich = int(input("Enter Your Choice : "))
                    if iich == 1:
                        purchase_add()
                    elif iich == 2:
                        purchase_list()
                    elif iich == 3:
                        break
                    else:
                        print("Enter Valid Choice!!")
            elif ich == 3:
                while True:
                    print("1.Add", "2.List", "3.Back", sep='\t')
                    iich = int(input("Enter Your Choice : "))
                    if iich == 1:
                        payment_add()
                    elif iich == 2:
                        payment_list()
                    elif iich == 3:
                        break
                    else:
                        print("Enter Valid Choice!!")
```

```

elif ich == 4:
    while True:
        print("1.Add", "2.List", "3.Back", sep='\t')
        iich = int(input("Enter Your Choice : "))
        if iich == 1:
            receipt_add()
        elif iich == 2:
            receipt_list()
        elif iich == 3:
            break
        else:
            print("Enter Valid Choice!!")
elif ich == 5:
    ledger()
elif ich == 6:
    cash_book()
elif ich == 7:
    break
else:
    print("Enter Valid Option")

```

## inventory.py

```

def inventory_call():
    from production import prd_add, prd_list
    from closing_stock import closing_stock
    while True:
        print("1.Production    2.Closing Stock    3.Back")
        ich = int(input("Enter Your choice : "))
        if ich==1:
            while True:
                print("1.Add","2.List","3.Back")
                iich = int(input("Enter Your choice : "))
                if iich==1:
                    prd_add()
                elif iich==2:
                    prd_list()
                elif iich==3:
                    break
                else:
                    print("Enter Valid Choice!")
            elif ich==2:
                closing_stock()
            elif ich==3:
                break
        else:
            print("Enter Valid Choice!")

```





```

elif mch==3:
    break
else:
    print("Enter Valid Choice!!")

```

## sales.py

```

def sales():
    from sql import sql
    query = ("Create Table IF NOT EXISTS Sales (Date date NOT NULL,\
                                                    VCH_NO Integer NOT NULL,\
                                                    Party_Name varchar(30) NOT NULL,\
                                                    Item_Name varchar(30) NOT NULL,\
                                                    Quantity integer NOT NULL,\
                                                    Unit varchar(10) NOT NULL,\
                                                    Price integer NOT NULL,\
                                                    Amount integer NOT NULL)")

    sql(query)
    print("Successfully Loaded Sales Table!")
def sales_add():
    from datetime import datetime
    from datetime_entry import input_date
    from vch_no_generate import generate_sales_vch
    sales()
    while True:
        user_input_date = input_date()
        if user_input_date:
            # Converting the validated date to "yyyy-mm-dd" format
            formatted_date = user_input_date.strftime("%Y-%m-%d")
            break
        else:
            print("Invalid date. Please try again.")
    print("Date:-",user_input_date)
    vch_number = generate_sales_vch()
    print("Voucher Number:", vch_number)
    from sql import sql
    from party_menu import PartyNameEntryForm
    party_form = PartyNameEntryForm()
    party_form.party_name_entry_form()
    selected_party = party_form.selected_party
    if selected_party is not None:
        print("Party:-",selected_party)
    while True:
        from item_menu import ItemNameEntryForm
        # Create an instance of the ItemNameEntryForm class
        item_form = ItemNameEntryForm()
        # Call the item_name_entry_form method to display the form
        item_form.item_name_entry_form()
        # Retrieve the selected item after the form is closed
        selected_item = item_form.selected_item
        # Check if an item was selected
        if selected_item is not None:
            print("Selected item:", selected_item)
        else:
            print("No item selected.")
            break
    quantity = float(input("Enter Quantity: "))
    unit=input("Enter Unit : ")
    price = float(input("Enter Price: "))
    amount=price*quantity

```

```

from sql import sql
insert_query="INSERT INTO Sales VALUES('{}','{}','{}','{}','{}','{}','{}').format(user_input_date,\
vch_number,selected_party,selected_item,quantity,unit,price,amount)
sql(insert_query)
ledger_query="INSERT INTO {} (DrDate, DrParticulars, DrAmount)Values('{}',\
'Sales',{}).format(selected_party,user_input_date,amount)
sql(ledger_query)
stk_query="INSERT INTO {} (OutDate,ParticularsOut,QuantityOut,UnitOut,PriceOut, AmountOut)\
Values('{}','Sales',{},'{}','{}').format(selected_item,user_input_date,quantity,unit,price,amount)
sql(stk_query)
con=input("Do You Want To ADD more Items?(y,n): ")
if con=='n':
    break
def sales_list():
    sales()
    import mysql.connector as sql
    mydb = sql.connect(host='localhost', user='root', passwd='1234', database='AI')
    mycursor = mydb.cursor()
    mycursor.execute("SELECT * FROM Sales")
    res = mycursor.fetchall()
    # Get the column names
    column_names = ['Sales_Date', 'VCH_NO', 'Party_Name', 'Item_Name','Quantity','Unit','Price','Amount']
    # Calculate the max length for each column
    max_lengths = [max(len(column_names[i]), max(len(str(row[i])) for row in res))
                    for i in range(len(column_names))]
    # Increase the width for better alignment
    max_lengths = [max_length + 4 for max_length in max_lengths]
    # Create the top border
    border = "+" + "+".join("-" * (length + 2) for length in max_lengths) + "+"
    # Create the column names row
    column_row = "|" + "|".join(column_names[i].center(max_lengths[i])
                                for i in range(len(column_names))) + "|"
    # Create the separator row
    separator = "+" + "+".join("-" * (max_lengths[i] + 2)
                                for i in range(len(column_names))) + "+"
    # Print the table
    print(border)
    print(column_row)
    print(separator)
    for row in res:
        row_str = "|" + " | ".join(str(cell).center(max_lengths[i] + 2)
                                    for i, cell in enumerate(row)) + "|"
        print(row_str)
    # Create the bottom border
    bottom_border = "+" + "+".join("-" * (max_lengths[i] + 2)
                                    for i in range(len(column_names))) + "+"
    print(bottom_border)

```

# purchase.py

```
def purchase():
    from sql import sql
    query = ("Create Table IF NOT EXISTS purchase (Date date NOT NULL,\
                                                    VCH_NO Integer NOT NULL,\
                                                    Party_Name varchar(30) NOT NULL,\
                                                    Item_Name varchar(30) NOT NULL,\
                                                    Quantity integer NOT NULL,\
                                                    Unit varchar(10) NOT NULL,\
                                                    Price integer NOT NULL,\
                                                    Amount integer NOT NULL)")

    sql(query)
    print("Successfully Loaded purchase Table!")
def purchase_add():
    from datetime import datetime
    from datetime_entry import input_date
    from vch_no_generate import generate_purchase_vch
    purchase()
    while True:
        user_input_date = input_date()
        if user_input_date:
            # Converting the validated date to "yyyy-mm-dd" format
            formatted_date = user_input_date.strftime("%Y-%m-%d")
            break
        else:
            print("Invalid date. Please try again.")
    print("Date:-",user_input_date)
    vch_number = generate_purchase_vch()
    print("Voucher Number:", vch_number)
    from sql import sql
    from party_menu import PartyNameEntryForm
    party_form = PartyNameEntryForm()
    party_form.party_name_entry_form()
    selected_party = party_form.selected_party
    if selected_party is not None:
        print("Party:-",selected_party)
    while True:
        from item_menu import ItemNameEntryForm
        # Create an instance of the ItemNameEntryForm class
        item_form = ItemNameEntryForm()
        # Call the item_name_entry_form method to display the form
        item_form.item_name_entry_form()
        # Retrieve the selected item after the form is closed
        selected_item = item_form.selected_item
        # Check if an item was selected
        if selected_item is not None:
            print("Selected item:", selected_item)
        else:
            print("No item selected.")
            break
    quantity = float(input("Enter Quantity: "))
    unit=input("Enter Unit : ")
    price = float(input("Enter Price: "))
    amount=price*quantity
    from sql import sql
    insert_query="INSERT INTO purchase VALUES('{}','{}','{}','{}','{}','{}','{}').format(user_input_date,\
                                                    vch_number,selected_party,selected_item,quantity,unit,price,amount)

    sql(insert_query)
    ledger_query="INSERT INTO {} (CrDate, CrParticulars, CrAmount)Values('{}',\
                                                    'purchase',{}).format(selected_party,user_input_date,amount)

    sql(ledger_query)
```

```

stk_query="INSERT INTO {} (InDate,ParticularsIn,QuantityIn, UnitIn, PriceIn, AmountIn)\
Values ('{}','Purchase',{},'{}','{}').format(selected_item,user_input_date,quantity,unit,price,amount)
sql(stk_query)
con=input("Do You Want To ADD more Items?(y,n): ")
if con=='n':
    break

```

```

def purchase_list():
    import mysql.connector as sql
    mydb = sql.connect(host='localhost', user='root', passwd='1234', database='AI')
    mycursor = mydb.cursor()
    mycursor.execute("SELECT * FROM Purchase")
    res = mycursor.fetchall()
    # Get the column names
    column_names = ['Purchase_Date', 'VCH_NO', 'Party_Name', 'Item_Name', 'Quantity', 'Unit', 'Price', 'Amount']
    # Calculate the max length for each column
    max_lengths = [max(len(column_names[i]), max(len(str(row[i])) for row in res))
                    for i in range(len(column_names))]
    # Increase the width for better alignment
    max_lengths = [max_length + 4 for max_length in max_lengths]
    # Create the top border
    border = "+" + "+".join("-" * (length + 2) for length in max_lengths) + "+"
    # Create the column names row
    column_row = "|" + "|".join(column_names[i].center(max_lengths[i])
                                for i in range(len(column_names))) + "|"
    # Create the separator row
    separator = "+" + "+".join("-" * (max_lengths[i] + 2)
                                for i in range(len(column_names))) + "+"
    # Print the table
    print(border)
    print(column_row)
    print(separator)
    for row in res:
        row_str = "|" + " | ".join(str(cell).center(max_lengths[i] + 2)
                                    for i, cell in enumerate(row)) + "|"
        print(row_str)
    # Create the bottom border
    bottom_border = "+" + "+".join("-" * (max_lengths[i] + 2)
                                    for i in range(len(column_names))) + "+"
    print(bottom_border)

```

## payment.py

```

def payment():
    from sql import sql
    query = ("Create Table IF NOT EXISTS Payment (Date date NOT NULL,\
                                                    VCH_NO Integer NOT NULL,\
                                                    Party_Name varchar(30) NOT NULL,\
                                                    Amount integer NOT NULL)")

    sql(query)
    print("Successfully Loaded Payment's Table!")
def payment_add():
    from cash_book import cashbook_load
    from datetime import datetime
    from datetime_entry import input_date
    from vch_no_generate import generate_payment_vch
    payment()
    cashbook_load()

```

```
user_input_date = input_date()  
if user_input_date:  
# Converting the validated date to "yyyy-mm-dd" format  
    formatted_date = user_input_date.strftime("%Y-%m-%d")  
    break  
else:  
    print("Invalid date. Please try again.")
```

```
sql(insert_query)
ledger_query="INSERT INTO {} (DrDate, DrParticulars, DrAmount)Values('{}',\n                'Payment',{})".format(selected_party,user_input_date,amount)
sql(ledger_query)
cashbk_query="INSERT INTO CashBook (CrDate, CrParticulars, CrAmount)\n                VALUES ('{}','{}',{})".format(user_input_date,selected_party,amount)
sql(cashbk_query)
```

```
payment()
import mysql.connector as sql
mydb = sql.connect(host='localhost', user='root', passwd='1234', database='AI')
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM Payment")
res = mycursor.fetchall()

# Get the column names
column_names = ['Payment_Date', 'VCH_NO', 'Party_Name', 'Amount']

# Calculate the max length for each column
max_lengths = [max(len(column_names[i]), max(len(str(row[i])) for row in res))
               for i in range(len(column_names))]

# Increase the width for better alignment
max_lengths = [max_length + 4 for max_length in max_lengths]

# Create the top border
border = "+" + "+".join("-" * (length + 2) for length in max_lengths) + "+"

# Create the column names row
column_row = "|" + "|".join(column_names[i].center(max_lengths[i])
                           for i in range(len(column_names))) + "|"

# Create the separator row
separator = "+" + "+".join("-" * (max_lengths[i] + 2)
                          for i in range(len(column_names))) + "+"

# Print the table
print(border)
print(column_row)
print(separator)
```



```

for row in res:
    row_str = "|" + " | ".join(str(cell).center(max_lengths[i] + 2)
                                for i, cell in enumerate(row)) + "|"
    print(row_str)
# Create the bottom border
bottom_border = "+" + "+".join("-" * (max_lengths[i] + 2)
                                for i in range(len(column_names))) + "+"
print(bottom_border)

```

## receipt.py

```

def receipt():
    from sql import sql
    query = ("Create Table IF NOT EXISTS Receipt (Date date NOT NULL,\
                                                    VCH_NO Integer NOT NULL,\
                                                    Party_Name varchar(30) NOT NULL,\
                                                    Amount integer NOT NULL)")

    sql(query)
    print("Successfully Loaded Receipt's Table!")

def receipt_add():
    from datetime import datetime
    from datetime_entry import input_date
    from vch_no_generate import generate_receipt_vch
    from cash_book import cashbook_load

    cashbook_load()
    receipt()
    while True:
        user_input_date = input_date()
        if user_input_date:
            # Converting the validated date to "yyyy-mm-dd" format
            formatted_date = user_input_date.strftime("%Y-%m-%d")
            break
        else:
            print("Invalid date. Please try again.")
    print("Date:-",user_input_date)
    vch_number = generate_receipt_vch()
    print("Voucher Number:", vch_number)
    from sql import sql
    from party_menu import PartyNameEntryForm
    party_form = PartyNameEntryForm()
    party_form.party_name_entry_form()
    selected_party = party_form.selected_party
    if selected_party is not None:
        print("Party:-",selected_party)
    amount=int(input("Enter Amount : "))
    from sql import sql
    insert_query="INSERT INTO Receipt Values('{0}',{0},{0},{0})".format(user_input_date,\
                                                                    vch_number,selected_party,amount)

    sql(insert_query)
    ledger_query="INSERT INTO {} (CrDate, CrParticulars, CrAmount)Values('{0}',\
                                'receipt',{0})".format(selected_party,user_input_date,amount)

    sql(ledger_query)
    cashbk_query="INSERT INTO CashBook (DrDate, DrParticulars, DrAmount)\
                VALUES ('{0}','{0},{0})".format(user_input_date,selected_party,amount)
    sql(cashbk_query)

```

```

def receipt_list():
    receipt()
    import mysql.connector as sql
    mydb = sql.connect(host='localhost', user='root', passwd='1234', database='AI')
    mycursor = mydb.cursor()
    mycursor.execute("SELECT * FROM Receipt")
    res = mycursor.fetchall()
    # Get the column names
    column_names = ['Receipt_Date', 'VCH_NO', 'Party_Name', 'Amount']
    # Calculate the max length for each column
    max_lengths = [max(len(column_names[i]), max(len(str(row[i])) for row in res))
                    for i in range(len(column_names))]
    # Increase the width for better alignment
    max_lengths = [max_length + 4 for max_length in max_lengths]
    # Create the top border
    border = "+" + "+".join("-" * (length + 2) for length in max_lengths) + "+"
    # Create the column names row
    column_row = "|" + "|".join(column_names[i].center(max_lengths[i])
                                for i in range(len(column_names))) + "|"
    # Create the separator row
    separator = "+" + "+".join("-" * (max_lengths[i] + 2)
                                for i in range(len(column_names))) + "+"
    # Print the table
    print(border)
    print(column_row)
    print(separator)
    for row in res:
        row_str = "|" + " | ".join(str(cell).center(max_lengths[i] + 2)
                                   for i, cell in enumerate(row)) + "|"
        print(row_str)
    # Create the bottom border
    bottom_border = "+" + "+".join("-" * (max_lengths[i] + 2)
                                    for i in range(len(column_names))) + "+"
    print(bottom_border)

```

## ledger.py

```

def ledger():
    import mysql.connector as sql
    from party_menu import PartyNameEntryForm
    party_form = PartyNameEntryForm()
    party_form.party_name_entry_form()
    selected_party = party_form.selected_party
    if selected_party is not None:
        print("Party:-", selected_party)
        mydb = sql.connect(host='localhost', user='root', passwd='1234', database='AI')
        mycursor = mydb.cursor()
        # Fetch data for the first table
        mycursor.execute("SELECT DrDate, DrParticulars, DrAmount FROM {} \
                          WHERE DrDate IS NOT NULL AND DrParticulars IS NOT NULL AND \
                          DrAmount IS NOT NULL".format(selected_party))
        res1 = mycursor.fetchall()
        # Get the column names for the first table
        column_names1 = ["Debit_Date", "DrParticulars", "DrAmount"]
        # Fetch data for the second table
        mycursor.execute("SELECT CrDate, CrParticulars, CrAmount FROM {} \
                          WHERE CrDate IS NOT NULL AND CrParticulars IS NOT NULL AND \
                          CrAmount IS NOT NULL".format(selected_party))
        res2 = mycursor.fetchall()

```



```

# Get the column names for the second table
column_names2 = ["Credit_Date", "CrParticulars", "CrAmount"]

# Calculate the max length for each column for both tables
max_lengths1 = [max(len(column_names1[i]), max(len(str(row[i])) for row in res1))
                 for i in range(len(column_names1))]
max_lengths2 = [max(len(column_names2[i]), max(len(str(row[i])) for row in res2))
                 for i in range(len(column_names2))]

# Increase the width for better alignment
max_lengths1 = [max_length + 2 for max_length in max_lengths1]
max_lengths2 = [max_length + 2 for max_length in max_lengths2]
# Determine the maximum number of rows between the two tables
max_rows = max(len(res1), len(res2))

# Create the top border
border = "+" + "+".join("-" * (length + 2) for length in max_lengths1 + max_lengths2) + "+"
# Create the column names row for the first table
column_row1 = "|" + "|".join(column_names1[i].center(max_lengths1[i])
                             for i in range(len(column_names1))) + "|"
# Create the column names row for the second table
column_row2 = "|" + "|".join(column_names2[i].center(max_lengths2[i])
                             for i in range(len(column_names2))) + "|"

# Print the table headers
print(border)
print(column_row1 + column_row2)
# Create the separator row for the first table
separator1 = "+" + "+".join("-" * (max_lengths1[i] + 2)
                             for i in range(len(column_names1))) + "+"
# Create the separator row for the second table
separator2 = "+" + "+".join("-" * (max_lengths2[i] + 2)
                             for i in range(len(column_names2))) + "+"

# Print the table separators
print(separator1 + separator2)
# Print the rows for both tables
for i in range(max_rows):
# Print rows for the first table if available
    if i < len(res1):
        row_str1 = "|" + " | ".join(str(cell).center(max_lengths1[j])
                                     for j, cell in enumerate(res1[i])) + "|"
    else:
# Print empty cells if the first table has fewer rows
        row_str1 = "|" + " " * (max_lengths1[0] + max_lengths1[1] + max_lengths1[2] - 2) + "|"
# Print rows for the second table if available
    if i < len(res2):
        row_str2 = "|" + " | ".join(str(cell).center(max_lengths2[j])
                                     for j, cell in enumerate(res2[i])) + "|"
    else:
# Print empty cells if the second table has fewer rows
        row_str2 = "|" + " " * (max_lengths2[0] + max_lengths2[1] + max_lengths2[2] - 2) + "|"
    print(row_str1 + row_str2)

# Create the bottom border
bottom_border = "+" + "+".join("-" * (max_lengths1[i] + 2) for i in range(len(column_names1))) + "+"
bottom_border += "+" + "+".join("-" * (max_lengths2[i] + 2) for i in range(len(column_names2))) + "+"
print(bottom_border)

# Calculate total debit amount
dr_amt = "SELECT SUM(DrAmount) FROM {}".format(selected_party)
mycursor.execute(dr_amt)
drtotal = mycursor.fetchone()[0]
dramt = float(drtotal) if drtotal is not None else 0

```

```

# Calculate total credit amount
cr_amt = "SELECT SUM(CrAmount) FROM {}".format(selected_party)
mycursor.execute(cr_amt)
crtotal = mycursor.fetchone()[0]
cramt = float(crtotal) if crttotal is not None else 0

if dramt < cramt:
    print("Account Balance: ₹",cramt-dramt,"Dr.")
elif dramt > cramt:
    print("Account Balance: ₹",dramt-cramt,"Cr.")

```

## cash\_book.py

```

def cashbook_load():
    from sql import sql
    query = ("Create Table IF NOT EXISTS CashBook (DrDate date ,\
                                                    DrParticulars varchar(30) ,\
                                                    DrAmount integer ,\
                                                    CrDate date ,\
                                                    CrParticulars varchar(30) ,\
                                                    CrAmount integer )")

    sql(query)
    print("Successfully Loaded Cash Book!")
def cash_book():
    cashbook_load()
    # Fetch data for the first table
    import mysql.connector as sql
    mydb=sql.connect(host='localhost',user='root',passwd='1234',database='AI')
    mycursor=mydb.cursor()

    # Fetch data for the first table
    mycursor.execute("SELECT DrDate, DrParticulars, DrAmount FROM CashBook\
                      WHERE DrDate IS NOT NULL AND DrParticulars IS NOT NULL AND \
                      DrAmount IS NOT NULL")

    res1 = mycursor.fetchall()

    # Get the column names for the first table
    column_names1 = ["Debit_Date", "DrParticulars", "DrAmount"]

    # Fetch data for the second table
    mycursor.execute("SELECT CrDate, CrParticulars, CrAmount FROM CashBook\
                      WHERE CrDate IS NOT NULL AND CrParticulars IS NOT NULL AND \
                      CrAmount IS NOT NULL")

    res2 = mycursor.fetchall()

    # Get the column names for the second table
    column_names2 = ["Credit_Date", "CrParticulars", "CrAmount"]

    # Calculate the max length for each column for both tables
    max_lengths1 = [max(len(column_names1[i]), max(len(str(row[i])) for row in res1))
                     for i in range(len(column_names1))]
    max_lengths2 = [max(len(column_names2[i]), max(len(str(row[i])) for row in res2))
                     for i in range(len(column_names2))]

    # Increase the width for better alignment
    max_lengths1 = [max_length + 2 for max_length in max_lengths1]
    max_lengths2 = [max_length + 2 for max_length in max_lengths2]

    # Determine the maximum number of rows between the two tables
    max_rows = max(len(res1), len(res2))

    # Create the top border
    border = "+" + "+" * (length + 2) for length in max_lengths1 + max_lengths2 + "+"

    # Create the column names row for the first table
    column_row1 = "|" + "|".join(column_names1[i].center(max_lengths1[i])
                                  for i in range(len(column_names1))) + "|"

```

```

# Create the column names row for the second table
column_row2 = "|" + "|".join(column_names2[i].center(max_lengths2[i])
                             for i in range(len(column_names2))) + "|"

# Print the table headers
print(border)
print(column_row1 + column_row2)

# Create the separator row for the first table
separator1 = "+" + "+".join("-" * (max_lengths1[i] + 2)
                             for i in range(len(column_names1))) + "+"

# Create the separator row for the second table
separator2 = "+" + "+".join("-" * (max_lengths2[i] + 2)
                             for i in range(len(column_names2))) + "+"

# Print the table separators
print(separator1 + separator2)

# Find the maximum length of rows between both tables
max_rows = max(len(res1), len(res2))

# Print the rows for both tables
for i in range(max_rows):
    # Print rows for the first table if available
    if i < len(res1):
        row_str1 = "|" + "|".join(str(cell).center(max_lengths1[j])
                                   for j, cell in enumerate(res1[i])) + "|"
    else:
        # Print empty cells if the first table has fewer rows
        row_str1 = "|" + " " * (max_lengths1[0] + max_lengths1[1] + max_lengths1[2] - 2) + "|"

    # Print rows for the second table if available
    if i < len(res2):
        row_str2 = "|" + "|".join(str(cell).center(max_lengths2[j])
                                   for j, cell in enumerate(res2[i])) + "|"
    else:
        # Print empty cells if the second table has fewer rows
        row_str2 = "|" + " " * (max_lengths2[0] + max_lengths2[1] + max_lengths2[2] - 2) + "|"

    print(row_str1 + row_str2)

# Create the bottom border
bottom_border = "+" + "+".join("-" * (max_lengths1[i] + 2) for i in range(len(column_names1))) + "+"
bottom_border += "+" + "+".join("-" * (max_lengths2[i] + 2) for i in range(len(column_names2))) + "+"
print(bottom_border)

# Calculate total debit amount
dr_amt = "SELECT SUM(DrAmount) FROM CashBook"
mycursor.execute(dr_amt)
drtotal = mycursor.fetchone()[0]
dramt = float(drtotal) if drtotal is not None else 0

# Calculate total credit amount
cr_amt = "SELECT SUM(CrAmount) FROM CashBook"
mycursor.execute(cr_amt)
crtotal = mycursor.fetchone()[0]
cramt = float(crtotal) if crttotal is not None else 0

if dramt < cramt:
    print("Cash Balance: ₹", cramt - dramt, "Dr.")
elif dramt > cramt:
    print("Cash Balance: ₹", dramt - cramt, "Cr.")

```

# production.py

```
def production():
    from sql import sql
    query = ("CREATE TABLE IF NOT EXISTS Production ("
        "Date DATE ,"
        "Item_consumed VARCHAR(30) ,"
        "Qnt_consumed INTEGER ,"
        "Unit_consumed VARCHAR(10) ,"
        "Item_generated VARCHAR(30),"
        "Qnt_generated INTEGER ,"
        "Unit_generated VARCHAR(10) );")
    sql(query)
    print("Successfully Loaded Production Table!")
def prd_add():
    from datetime import datetime
    from datetime_entry import input_date
    production()
    while True:
        user_input_date = input_date()
        if user_input_date:
            # Converting the validated date to "yyyy-mm-dd" format
            formatted_date = user_input_date.strftime("%Y-%m-%d")
            break
        else:
            print("Invalid date. Please try again.")
    print("Date:-",user_input_date)
    #Item Consumed Details
    while True:
        print("Select Consumption item!")
        from item_menu import ItemNameEntryForm
        # Create an instance of the ItemNameEntryForm class
        item_form = ItemNameEntryForm()
        # Call the item_name_entry_form method to display the form
        item_form.item_name_entry_form()
        # Retrieve the selected item after the form is closed
        selected_item = item_form.selected_item
        # Check if an item was selected
        if selected_item is not None:
            print("Selected item:", selected_item)
        else:
            print("No item selected.")
            break
    quantity = float(input("Enter Quantity: "))
    unit=input("Enter Unit : ")
    from sql import sql
    insert_query="INSERT INTO Production (Date,Item_consumed,Qnt_consumed,Unit_consumed)\
        VALUES('{}','{}',{},{})".format(user_input_date,selected_item,quantity,unit)
    sql(insert_query)
    price=1
    amount=1
    stk_query="INSERT INTO {} (OutDate,ParticularsOut,QuantityOut, UnitOut, PriceOut, AmountOut)\
        Values ('{}','Production',{},{},{})".format(selected_item,user_input_date,quantity,unit,price,amount)
    sql(stk_query)
    con=input("Do You Want To ADD more Consumption Items?{y,n}: ")
    if con=='n':
        break
```

## #Item Generated Details

while True:

print("Select Generated item!")

from item\_menu import ItemNameEntryForm

# Create an instance of the ItemNameEntryForm class

item\_form = ItemNameEntryForm()

# Call the item\_name\_entry\_form method to display the form

item\_form.item\_name\_entry\_form()

# Retrieve the selected item after the form is closed

selected\_item = item\_form.selected\_item

# Check if an item was selected

if selected\_item is not None:

print("Selected item:", selected\_item)

else:

print("No item selected.")

break

quantity = float(input("Enter Quantity: "))

unit=input("Enter Unit : ")

from sql import sql

insert\_query="INSERT INTO Production (Item\_generated,Qnt\_generated,Unit\_generated)\nVALUES('{}','{}','{}').format(selected\_item,quantity,unit)"

sql(insert\_query)

price=1

amount=1

stk\_query="INSERT INTO {} (InDate,ParticularsIn,QuantityIn, UnitIn, PriceIn, AmountIn)\nValues ('{}','Production',{},'{}','{}').format(selected\_item,user\_input\_date,quantity,unit,price,amount)"

sql(stk\_query)

con=input("Do You Want To ADD more Generated Items?{y,n}: ")

if con=='n':

break

break

def prd\_list():

production()

# Fetch data for the first table

import mysql.connector as sql

mydb=sql.connect(host='localhost',user='root',passwd='1234',database='AI')

mycursor=mydb.cursor()

mycursor.execute("SELECT Date, Item\_consumed, Qnt\_consumed, Unit\_consumed \nFROM Production WHERE Date IS NOT NULL AND Item\_consumed IS NOT NULL \nAND Qnt\_consumed IS NOT NULL AND Unit\_consumed IS NOT NULL")

res1 = mycursor.fetchall()

# Get the column names for the first table

column\_names1 = ["Date", "ItemCons.", "QntCons.", "UnitCons."]

# Fetch data for the second table

mycursor.execute("SELECT Item\_generated, Qnt\_generated,Unit\_generated \nFROM Production WHERE Item\_generated IS NOT NULL AND \nQnt\_generated IS NOT NULL AND Unit\_generated IS NOT NULL")

res2 = mycursor.fetchall()

# Get the column names for the second table

column\_names2 = ["ItemGenrt.", "QntGenrt.", "UnitGenrt."]

# Calculate the max length for each column for both tables

max\_lengths1 = [max(len(column\_names1[i]), max(len(str(row[i])) for row in res1))\nfor i in range(len(column\_names1))]

max\_lengths2 = [max(len(column\_names2[i]), max(len(str(row[i])) for row in res2))\nfor i in range(len(column\_names2))]

# Increase the width for better alignment

max\_lengths1 = [max\_length + 2 for max\_length in max\_lengths1]

max\_lengths2 = [max\_length + 2 for max\_length in max\_lengths2]

# Determine the maximum number of rows between the two tables

max\_rows = max(len(res1), len(res2))

# Create the top border

border = "+" + "+".join("-" \* (length + 2) for length in max\_lengths1 + max\_lengths2) + "+"



```

# Create the column names row for the first table
column_row1 = "|" + "|".join(column_names1[i].center(max_lengths1[i])
                             for i in range(len(column_names1))) + "|"

# Create the column names row for the second table
column_row2 = "|" + "|".join(column_names2[i].center(max_lengths2[i])
                             for i in range(len(column_names2))) + "|"

# Print the table headers
print(border)
print(column_row1 + column_row2)

# Create the separator row for the first table
separator1 = "+" + "+".join("-" * (max_lengths1[i] + 2)
                             for i in range(len(column_names1))) + "+"

# Create the separator row for the second table
separator2 = "+" + "+".join("-" * (max_lengths2[i] + 2)
                             for i in range(len(column_names2))) + "+"

# Print the table separators
print(separator1 + separator2)

# Find the maximum length of rows between both tables
max_rows = max(len(res1), len(res2))

# Print the rows for both tables
for i in range(max_rows):
    # Print rows for the first table if available
    if i < len(res1):
        row_str1 = "|" + " | ".join(str(cell).center(max_lengths1[j])
                                     for j, cell in enumerate(res1[i])) + "|"

    else:
        # Print empty cells if the first table has fewer rows
        row_str1 = "|" + " " * (max_lengths1[0] + max_lengths1[1] + max_lengths1[2] - 2) + "|"

    # Print rows for the second table if available
    if i < len(res2):
        row_str2 = " | " + "|".join(str(cell).center(max_lengths2[j])
                                    for j, cell in enumerate(res2[i])) + "|"

    else:
        # Print empty cells if the second table has fewer rows
        row_str2 = "|" + " " * (max_lengths2[0] + max_lengths2[1] + max_lengths2[2] - 2) + "|"

    print(row_str1 + row_str2)

# Create the bottom border
bottom_border = "+" + "+".join("-" * (max_lengths1[i] + 2) for i in range(len(column_names1))) + "+"
bottom_border += "+" + "+".join("-" * (max_lengths2[i] + 2) for i in range(len(column_names2))) + "+"
print(bottom_border)

```

## closing\_stock.py

```

def closing_stock():
    from item_menu import ItemNameEntryForm
    # Create an instance of the ItemNameEntryForm class
    item_form = ItemNameEntryForm()
    # Call the item_name_entry_form method to display the form
    item_form.item_name_entry_form()
    # Retrieve the selected item after the form is closed
    selected_item = item_form.selected_item
    # Check if an item was selected
    if selected_item is not None:
        print("Selected item:", selected_item)
    else:
        print("No item selected.")
    # Fetch data for the first table
    import mysql.connector as sql
    mydb = sql.connect(host='localhost', user='root', passwd='1234', database='AI')
    mycursor = mydb.cursor()

```

```

mycursor.execute("SELECT InDate, ParticularsIn, QuantityIn, UnitIn, PriceIn, AmountIn FROM {} \
WHERE InDate IS NOT NULL AND ParticularsIn IS NOT NULL AND UnitIn IS NOT NULL \
AND QuantityIn IS NOT NULL AND PriceIn IS NOT NULL AND AmountIn IS NOT NULL".format(selected_item))
res1 = mycursor.fetchall()
# Get the column names for the first table
column_names1 = ["In_Date", "PrtclsIn", "QntIn", "UnitIn", "PriceIn", "AmountIn"]
# Fetch data for the second table
mycursor.execute("SELECT OutDate, ParticularsOut, QuantityOut, UnitOut, PriceOut, AmountOut FROM {} \
WHERE OutDate IS NOT NULL AND ParticularsOut IS NOT NULL AND UnitOut IS NOT NULL \
AND QuantityOut IS NOT NULL AND PriceOut IS NOT NULL AND AmountOut IS NOT NULL".format(selected_item))
res2 = mycursor.fetchall()
# Get the column names for the second table
column_names2 = ["Out_Date", "PrtclsOut", "QntOut", "UnitOut", "PriceOut", "AmountOut"]
# Check if res2 is not empty before calculating max_lengths2
if res2:
    max_lengths2 = [max(len(column_names2[i]), max(len(str(row[i])) for row in res2)) + 2
                     for i in range(len(column_names2))]
else:
    max_lengths2 = [len(column_name) + 2 for column_name in column_names2]
# Calculate the max length for each column for both tables
max_lengths1 = [max(len(column_names1[i]), max(len(str(row[i])) for row in res1))
                 for i in range(len(column_names1))]
# Increase the width for better alignment
max_lengths1 = [max_length + 2 for max_length in max_lengths1]
max_lengths2 = [max_length + 2 for max_length in max_lengths2]
# Determine the maximum number of rows between the two tables
max_rows = max(len(res1), len(res2))
# Create the top border
border = "+" + "+" * (length + 2) for length in max_lengths1 + max_lengths2 + "+"
# Create the column names row for the first table
column_row1 = "|" + "|".join(column_names1[i].center(max_lengths1[i])
                             for i in range(len(column_names1))) + "|"
# Create the column names row for the second table
column_row2 = "|" + "|".join(column_names2[i].center(max_lengths2[i])
                             for i in range(len(column_names2))) + "|"
# Print the table headers
print(border)
print(column_row1 + column_row2)
# Create the separator row for the first table
separator1 = "+" + "+" * (max_lengths1[i] + 2)
               for i in range(len(column_names1))) + "+"
# Create the separator row for the second table
separator2 = "+" + "+" * (max_lengths2[i] + 2)
               for i in range(len(column_names2))) + "+"
# Print the table separators
print(separator1 + separator2)
# Find the maximum length of rows between both tables
max_rows = max(len(res1), len(res2))
# Print the rows for both tables
for i in range(max_rows):
    # Print rows for the first table if available
    if i < len(res1):
        row_str1 = "|" + "|".join(str(cell).center(max_lengths1[j])
                                   for j, cell in enumerate(res1[i])) + "|"
    else:
        # Print empty cells if the first table has fewer rows
        row_str1 = "|" + " " * (max_lengths1[0] + max_lengths1[1] + max_lengths1[2] - 2) + "|"
    # Print rows for the second table if available
    if i < len(res2):
        row_str2 = "|" + "|".join(str(cell).center(max_lengths2[j])
                                   for j, cell in enumerate(res2[i])) + "|"
    else:
        # Print empty cells if the second table has fewer rows
        row_str2 = "|" + " " * (max_lengths2[0] + max_lengths2[1] + max_lengths2[2] - 2) + "|"

    print(row_str1 + row_str2)
# Create the bottom border
bottom_border = "+" + "+" * (max_lengths1[i] + 2) for i in range(len(column_names1))) + "+"
bottom_border += "+" + "+" * (max_lengths2[i] + 2) for i in range(len(column_names2))) + "+"
print(bottom_border)

```

```

# Calculate Item IN
itm_in = "SELECT SUM(QuantityIn) FROM {}".format(selected_item)
mycursor.execute(itm_in)
intotal = mycursor.fetchone()[0]
initm = float(intotal) if intotal is not None else 0
# Calculate Item OUT
itm_out = "SELECT SUM(QuantityOut) FROM {}".format(selected_item)
mycursor.execute(itm_out)
outtotal = mycursor.fetchone()[0]
outitm = float(outtotal) if outtotal is not None else 0

print("Closing Stock is : ", initm - outitm, "UNITS")

```

## party.py

```

def party_name():
    from sql import sql
    query = ("Create Table IF NOT EXISTS PartyName (Party varchar(30) NOT NULL Primary Key,\
            Address varchar(50) NOT NULL,\
            MobileNo integer NOT NULL)")

    sql(query)
    print("Successfully Loaded Party Table!")

def party_add():
    party_name()
    from sql import sql
    p_name=input("Please Enter Party Name : ")
    p_adrs=input("Please Enter Address : ")
    p_no=input("Please Enter Number : ")
    create_query=("Create Table IF NOT EXISTS {} (DrDate date ,\
            DrParticulars varchar(30) ,\
            DrAmount integer ,\
            CrDate date ,\
            CrParticulars varchar(30) ,\
            CrAmount integer )".format(p_name))

    insert_query = "Insert into PartyName values('{}','{}',{})".format(p_name, p_adrs, p_no)
    sql(create_query)
    sql(insert_query)
    print("Party Added Successfully!")

def party_modify():
    from sql import sql
    from party_menu import PartyNameEntryForm
    party_form = PartyNameEntryForm()
    party_form.party_name_entry_form()
    selected_party = party_form.selected_party
    if selected_party is not None:
        print("Party Found!")
        while True:
            qch = input("What You Want To Modify (Adrs\Mo): ")
            if qch == 'Adrs':
                address = input("Enter New Address: ")
                modify_query = "UPDATE PartyName SET Address='{}' WHERE Party='{}'".format(address, selected_party)
                sql(modify_query)
                print("Party details Updated Successfully!")
                break
            elif qch == 'Mo':
                mobile = int(input("Enter New Number: "))
                modify_query = "UPDATE PartyName SET MobileNo={} WHERE Party='{}'".format(mobile, selected_party)
                sql(modify_query)
                print("Party details Updated Successfully!")
                break
            else:
                print("Invalid Choice!!")
                print("Enter choice from this {Adrs or Mo}")
        else:
            print("No party selected.")

```



```

def party_delete():
    from sql import sql
    from party_menu import PartyNameEntryForm
    party_form = PartyNameEntryForm()
    party_form.party_name_entry_form()
    selected_party = party_form.selected_party
    if selected_party is not None:
        print("Party Found!")
        query1="DROP Table {}".format(selected_party)
        sql(query1)
        query2="DELETE From PartyName WHERE Party='{}' ".format(selected_party)
        sql(query2)

def party_list():
    import mysql.connector as sql
    mydb = sql.connect(host='localhost', user='root', passwd='1234', database='AI')
    mycursor = mydb.cursor()
    mycursor.execute("SELECT * FROM PartyName")
    res = mycursor.fetchall()
    # Get the column names
    column_names = ["PartyName", "Address","Mobile No."]
    # Calculate the max length for each column
    max_lengths = [max(len(column_names[i]), max(len(str(row[i])) for row in res))
                    for i in range(len(column_names))]
    # Increase the width for better alignment
    max_lengths = [max_length + 4 for max_length in max_lengths]
    # Create the top border
    border = "+" + "+".join("-" * (length + 2) for length in max_lengths) + "+"
    # Create the column names row
    column_row = "|" + "|".join(column_names[i].center(max_lengths[i])
                                for i in range(len(column_names))) + "|"
    # Create the separator row
    separator = "+" + "+".join("-" * (max_lengths[i] + 2)
                                for i in range(len(column_names))) + "+"
    # Print the table
    print(border)
    print(column_row)
    print(separator)
    for row in res:
        row_str = "|" + " | ".join(str(cell).center(max_lengths[i] + 2)
                                   for i, cell in enumerate(row)) + "|"
        print(row_str)
    # Create the bottom border
    bottom_border = "+" + "+".join("-" * (max_lengths[i] + 2)
                                    for i in range(len(column_names))) + "+"
    print(bottom_border)

```

item.py

```

def item_name():
    from sql import sql
    query = ("Create Table IF NOT EXISTS ItemName\
             (Item varchar(30) NOT NULL Primary Key,Unit varchar(6) NOT NULL)")
    sql(query)
    print("Successfully Loaded Item Table!")

def item_add():
    item_name()
    from sql import sql
    i_name=input("Please Enter Item Name : ")
    i_unit=input("Please Enter Unit : ")
    i_price=int(input("Please Enter Price : "))

```

```

create_query=("Create Table IF NOT EXISTS {} (InDate date,\
                                                    ParticularsIn varchar(30),\
                                                    QuantityIn Integer,\
                                                    UnitIn varchar(6) ,\
                                                    PriceIn integer ,\
                                                    AmountIn integer,\
                                                    OutDate date,\
                                                    ParticularsOut varchar(30),\
                                                    QuantityOut Integer,\
                                                    UnitOut varchar(6) ,\
                                                    PriceOut integer ,\
                                                    AmountOut integer )".format(i_name))

insert_query = "Insert into ItemName values('{}','{}').format(i_name, i_unit)
insert_queryy = "insert into {} (Unit) Values ('{}').format(i_name, i_unit)
sql(create_query)
sql(insert_query)
print("Item Added Successfully!")

```

```

def item_modify():
    item_name()
    from sql import sql
    from item_menu import ItemNameEntryForm
    item_form = ItemNameEntryForm()
    item_form.item_name_entry_form()
    selected_item = item_form.selected_item
    if selected_item is not None:
        print("Item Found!")
        qch = input("What You Want To Modify (Unit/Name): ")
        if qch == 'Unit':
            unit = input("Enter New Unit : ")
            modify_query = "UPDATE ItemName SET Unit='{}' \
                            WHERE item='{}'".format(unit, selected_item)
            sql(modify_query)
            print("Item details Updated Successfully!")
        elif qch == 'Name':
            name = input("Enter New Name : ")
            modify_query = "UPDATE itemname SET item='{}' \
                            WHERE item='{}'".format(name, selected_item)
            modify_queryy="Alter Table {} Rename {}".format(selected_item,name)
            sql(modify_query)
            print("Item details Updated Successfully!")
        else:
            print("Invalid Choice!!")
            print("Enter choice from this {Unit or Name}")
    else:
        print("No item selected.")

```

```

def item_list():
    item_name()
    import mysql.connector as sql
    mydb = sql.connect(host='localhost', user='root', passwd='1234', database='AI')
    mycursor = mydb.cursor()
    mycursor.execute("SELECT * FROM ItemName")
    res = mycursor.fetchall()
    # Get the column names
    column_names = ["ItemName", "Unit"]

```

```

# Calculate the max length for each column
max_lengths = [max(len(column_names[i]), max(len(str(row[i])) for row in res))
                for i in range(len(column_names))]

# Increase the width for better alignment
max_lengths = [max_length + 4 for max_length in max_lengths]

# Create the top border
border = "+" + "+".join("-" * (length + 2) for length in max_lengths) + "+"

# Create the column names row
column_row = "|" + "|".join(column_names[i].center(max_lengths[i])
                             for i in range(len(column_names))) + "|"

# Create the separator row
separator = "+" + "+".join("-" * (max_lengths[i] + 2)
                             for i in range(len(column_names))) + "+"

# Print the table
print(border)
print(column_row)
print(separator)
for row in res:
    row_str = "|" + " | ".join(str(cell).center(max_lengths[i] + 2)
                               for i, cell in enumerate(row)) + "|"

    print(row_str)

# Create the bottom border
bottom_border = "+" + "+".join("-" * (max_lengths[i] + 2)
                                for i in range(len(column_names))) + "+"

print(bottom_border)

```

```

def item_delete():
    from sql import sql
    from item_menu import ItemNameEntryForm
    item_form = ItemNameEntryForm()
    item_form.item_name_entry_form()
    selected_item = item_form.selected_item
    if selected_item is not None:
        print("Item Found!")
        delete_query="Delete FROM ItemName where item='{0}' ".format(selected_item)
        sql(delete_query)
        table_query="DROP Table {0}".format(selected_item)
        sql(table_query)
        print("Item Deleted Successfully")

```

## datetime\_entry.py

```

from datetime import datetime

def input_date():
    date_string = input("Enter Date (DD-MM-YYYY) : ")
    try:
        day, month, year = map(int, date_string.split('-'))
        user_date = datetime(year, month, day)
        return user_date
    except (ValueError, IndexError):
        print("Invalid format. Please use DD-MM-YYYY format.")
        return None

```

# item\_menu.py

```
import tkinter as tk
from tkinter import ttk
import mysql.connector
class ItemNameEntryForm:
    def __init__(self):
        self.selected_item = None
    def save_and_destroy(self):
        self.selected_item = self.item_combo.get()
        # Perform validation
        if not self.selected_item:
            print("Please select a item name.")
            return
        # Close the form
        self.app.destroy()
    def get_item_names(self):
        connection = mysql.connector.connect(host="localhost",user="root",passwd="1234",database="AI")
        cursor = connection.cursor()
        cursor.execute("SELECT item FROM ItemName")
        item_names = [row[0] for row in cursor.fetchall()]
        connection.close()
        return item_names
    def item_name_entry_form(self):
        self.app = tk.Tk()
        self.app.title("Item Selection")
        # item Name ComboBox
        item_label = tk.Label(self.app, text="Item Name:")
        item_label.pack()
        # Fetch item names from MySQL
        item_names = self.get_item_names()
        self.item_combo = ttk.Combobox(self.app, values=item_names)
        self.item_combo.pack()
        # Save Button
        save_button = tk.Button(self.app, text="OK", command=self.save_and_destroy)
        save_button.pack()
        self.app.mainloop()
```

# sql.py

```
def sql(query):
    import mysql.connector as sql
    mydb=sql.connect(host='localhost',user='root',passwd='1234',database='AI')
    mycursor=mydb.cursor()
    mycursor.execute(query)
    mydb.commit()
    print("Query Executed Successfully!")
```

# item\_menu.py

```
import tkinter as tk
from tkinter import ttk
import mysql.connector
class PartyNameEntryForm:
    def __init__(self):
        self.selected_party = None
    def save_and_destroy(self):
        self.selected_party = self.party_combo.get()
        # Perform validation
        if not self.selected_party:
            print("Please select a party name.")
            return
        # Close the form
        self.app.destroy()
    def get_party_names(self):
        connection = mysql.connector.connect(host="localhost",user="root",passwd="1234",database="AI")
        cursor = connection.cursor()
        cursor.execute("SELECT Party FROM PartyName")
        party_names = [row[0] for row in cursor.fetchall()]
        connection.close()
        return party_names
    def party_name_entry_form(self):
        self.app = tk.Tk()
        self.app.title("Party Selection")
        # Party Name ComboBox
        party_label = tk.Label(self.app, text="Party Name:")
        party_label.pack()
        # Fetch party names from MySQL
        party_names = self.get_party_names()
        self.party_combo = ttk.Combobox(self.app, values=party_names)
        self.party_combo.pack()
        # Save Button
        save_button = tk.Button(self.app, text="OK", command=self.save_and_destroy)
        save_button.pack()
        self.app.mainloop()
```

# vch\_no\_generate.py

```
def generate_sales_vch():
    import mysql.connector as mysql
    conn = mysql.connect(host="localhost",user="root",passwd="1234",database="AI")
    cursor = conn.cursor()
    cursor.execute("SELECT MAX(vch_no) FROM Sales")
    latest_voucher_number = cursor.fetchone()[0] or 0
    conn.close()
    # Incrementing the latest voucher number
    vch_number = latest_voucher_number + 1
    return vch_number
```

```
def generate_purchase_vch():
```

```
    import mysql.connector as mysql
```

```
    conn = mysql.connect(host="localhost",user="root",passwd="1234",database="AI")
```

```
    cursor = conn.cursor()
```

```
    cursor.execute("SELECT MAX(vch_no) FROM Sales")
```

```
    latest_voucher_number = cursor.fetchone()[0] or 0
```

```
    conn.close()
```

```
    # Incrementing the latest voucher number
```

```
    vch_number = latest_voucher_number + 1
```

```
    return vch_number
```

```
def generate_receipt_vch():
```

```
    import mysql.connector as mysql
```

```
    conn = mysql.connect(host="localhost",user="root",passwd="1234",database="AI")
```

```
    cursor = conn.cursor()
```

```
    cursor.execute("SELECT MAX(vch_no) FROM Sales")
```

```
    latest_voucher_number = cursor.fetchone()[0] or 0
```

```
    conn.close()
```

```
    # Incrementing the latest voucher number
```

```
    vch_number = latest_voucher_number + 1
```

```
    return vch_number
```

```
def generate_payment_vch():
```

```
    import mysql.connector as mysql
```

```
    conn = mysql.connect(host="localhost",user="root",passwd="1234",database="AI")
```

```
    cursor = conn.cursor()
```

```
    cursor.execute("SELECT MAX(vch_no) FROM Sales")
```

```
    latest_voucher_number = cursor.fetchone()[0] or 0
```

```
    conn.close()
```

```
    # Incrementing the latest voucher number
```

```
    vch_number = latest_voucher_number + 1
```

```
    return vch_number
```

# Output Screen

## Adding Party

\_\_\_\_\_Toh Kaise Hai APP Log!!!\_\_\_\_\_

\_\_\_\_\_Mini Accounting & Inventory Management Software\_\_\_\_\_

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 3

1.Party 2.Item 3.Back

Enter Your Choice : 1

1.Add 2.Modify 3.List 4.Back

Enter Your Choice : 1

Query Executed Successfully!

Successfully Loaded Party Table!

Please Enter Party Name : Jainil

Please Enter Address : Maninagar

Please Enter Number : 7990178960

Query Executed Successfully!

Query Executed Successfully!

Party Added Successfully!

## Viewing Parties

\_\_\_\_\_Toh Kaise Hai APP Log!!!\_\_\_\_\_

\_\_\_\_\_Mini Accounting & Inventory Management Software\_\_\_\_\_

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 3

1.Party 2.Item 3.Back

Enter Your Choice : 1

1.Add 2.Modify 3.List 4.Back

Enter Your Choice : 3

+-----+-----+-----+

| PartyName | Address | Mobile No. |

+-----+-----+-----+

| Jainil | Maninagar | 7990178960 |

| Riya | Maninagar | 9510614493 |

+-----+-----+-----+



# Adding Item

\_\_\_\_\_**Toh Kaise Hai APP Log!!!**\_\_\_\_\_

\_\_\_\_\_**Mini Accounting & Inventory Management Software**\_\_\_\_\_

**1.Accounting**

**2.Inventory**

**3.Masters**

**4.Exit**

**Choose An Option : 3**

**1.Party 2.Item 3.Back**

**Enter Your Choice : 2**

**1.Add 2.Modify 3.List 4.Back**

**Enter Your Choice : 1**

**Query Executed Successfully!**

**Successfully Loaded Item Table!**

**Please Enter Item Name : Item1**

**Please Enter Unit : KGS**

**Please Enter Price : 100**

**Query Executed Successfully!**

**Query Executed Successfully!**

**Item Added Successfully!**

# Viewing Items

\_\_\_\_\_**Toh Kaise Hai APP Log!!!**\_\_\_\_\_

\_\_\_\_\_**Mini Accounting & Inventory Management Software**\_\_\_\_\_

**1.Accounting**

**2.Inventory**

**3.Masters**

**4.Exit**

**Choose An Option : 3**

**1.Party 2.Item 3.Back**

**Enter Your Choice : 2**

**1.Add 2.Modify 3.List 4.Back**

**Enter Your Choice : 3**

**Query Executed Successfully!**

**Successfully Loaded Item Table!**

**+-----+-----+**

**| ItemName | Unit |**

**+-----+-----+**

**| Item1 | KGS |**

**| Item2 | PCS |**

**| Item3 | LTR |**

**| Item4 | NOS |**

**+-----+-----+**



# Adding Sales Voucher

\_\_\_\_\_Toh Kaise Hai APP Log!!!\_\_\_\_\_

\_\_\_\_\_Mini Accounting & Inventory Management Software\_\_\_\_\_

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 1

1.Add 2.List 3.Back

Enter Your Choice : 1

Query Executed Successfully!

Successfully Loaded Sales Table!

Enter Date (DD-MM-YYYY) : 1-4-2023

Date:- 2023-04-01 00:00:00

Voucher Number: 1

Party:- Jainil

Selected item: Item1

Enter Quantity: 15

Enter Unit : KGS

Enter Price: 100

Query Executed Successfully!

Query Executed Successfully!

Query Executed Successfully!

Do You Want To ADD more Items?{y,n}: n

# Viewing Sales Voucher

\_\_\_\_\_Toh Kaise Hai APP Log!!!\_\_\_\_\_

\_\_\_\_\_Mini Accounting & Inventory Management Software\_\_\_\_\_

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 1

1.Add 2.List 3.Back

Enter Your Choice : 2

Query Executed Successfully!

Successfully Loaded Sales Table!

Sales_Date	VCH_NO	Party_Name	Item_Name	Quantity	Unit	Price	Amount
2023-04-01	1	Jainil	Item1	15	KGS	100	1500

# Adding Purchase Voucher

Toh Kaise Hai APP Log!!!

Mini Accounting & Inventory Management Software

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 2

1.Add 2.List 3.Back

Enter Your Choice : 1

Query Executed Successfully!

Successfully Loaded purchase Table!

Enter Date (DD-MM-YYYY) : 1-4-2023

Date:- 2023-04-01 00:00:00

Voucher Number: 1

Party:- Riya

Selected item: Item1

Enter Quantity: 200

Enter Unit : KGS

Enter Price: 55

Query Executed Successfully!

Query Executed Successfully!

Query Executed Successfully!

Do You Want To ADD more Items?{y,n}: n

# Viewing Purchase Voucher

Toh Kaise Hai APP Log!!!

Mini Accounting & Inventory Management Software

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 2

1.Add 2.List 3.Back

Enter Your Choice : 2

Purchase_Date	VCH_NO	Party_Name	Item_Name	Quantity	Unit	Price	Amount
2023-04-01	1	Riya	Item1	200	KGS	55	11000

# Adding Payment Voucher

Toh Kaise Hai APP Log!!!

Mini Accounting & Inventory Management Software

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 3

1.Add 2.List 3.Back

Enter Your Choice : 1

Query Executed Successfully!

Successfully Loaded Payment's Table!

Query Executed Successfully!

Successfully Loaded Cash Book!

Enter Date (DD-MM-YYYY) : 3-4-2023

Date:- 2023-04-03 00:00:00

Voucher Number: 1

Party:- Riya

Enter Amount Paid : 9800

Query Executed Successfully!

Query Executed Successfully!

Query Executed Successfully!

# Viewing Payment Voucher

Toh Kaise Hai APP Log!!!

Mini Accounting & Inventory Management Software

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 3

1.Add 2.List 3.Back

Enter Your Choice : 2

Query Executed Successfully!

Successfully Loaded Payment's Table!

Payment_Date	VCH_NO	Party_Name	Amount
2023-04-03	1	Riya	9800

# Adding Receipt Voucher

\_\_\_\_\_Toh Kaise Hai APP Log!!!\_\_\_\_\_

\_\_\_\_\_Mini Accounting & Inventory Management Software\_\_\_\_\_

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 4

1.Add 2.List 3.Back

Enter Your Choice : 1

Query Executed Successfully!

Successfully Loaded Cash Book!

Query Executed Successfully!

Successfully Loaded Receipt's Table!

Enter Date (DD-MM-YYYY) : 2-4-2023

Date:- 2023-04-02 00:00:00

Voucher Number: 1

Party:- Jainil

Enter Amount : 10000

Query Executed Successfully!

Query Executed Successfully!

Query Executed Successfully!

# Viewing Receipt Voucher

\_\_\_\_\_Toh Kaise Hai APP Log!!!\_\_\_\_\_

\_\_\_\_\_Mini Accounting & Inventory Management Software\_\_\_\_\_

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 4

1.Add 2.List 3.Back

Enter Your Choice : 2

Query Executed Successfully!

Successfully Loaded Receipt's Table!

```
+-----+-----+-----+-----+
| Receipt_Date | VCH_NO | Party_Name | Amount |
+-----+-----+-----+-----+
| 2023-04-02  | 1      | Jainil    | 10000  |
+-----+-----+-----+-----+
```

# Ledger

\_\_\_\_\_Toh Kaise Hai APP Log!!!\_\_\_\_\_

\_\_\_\_\_Mini Accounting & Inventory Management Software\_\_\_\_\_

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 5

Party:- Jainil

```
+-----+-----+-----+-----+-----+-----+
| Debit_Date | DrPrtclrs | DrAmount || Credit_Date | CrPrtclrs | CrAmount |
+-----+-----+-----+-----+-----+-----+
| 2023-04-01 | Sales     | 1500    || 2023-04-02 | receipt   | 10000    |
+-----+-----+-----+-----+-----+-----+
```

Account Balance: ₹ 8500.0 Dr.

# Cash Book

\_\_\_\_\_Toh Kaise Hai APP Log!!!\_\_\_\_\_

\_\_\_\_\_Mini Accounting & Inventory Management Software\_\_\_\_\_

1.Accounting

2.Inventory

3.Masters

4.Exit

Choose An Option : 1

1.Sales 2.Purchase 3.Payment 4.Receipt 5.Ledger 6.Cash Book 7.Back

Enter Your choice : 6

Query Executed Successfully!

Successfully Loaded Cash Book!

```
+-----+-----+-----+-----+-----+-----+
| Debit_Date | DrPrtclrs|DrAmount ||Credit_Date |CrPrtclrs|CrAmount |
+-----+-----+-----+-----+-----+-----+
| 2023-04-02 | Jainil   | 10000   || 2023-04-03 | Riya     | 9800    |
+-----+-----+-----+-----+-----+-----+
```

Cash Balance: ₹ 200.0 Cr.

# Adding Production Voucher

**Toh Kaise Hai APP Log!!!**

**Mini Accounting & Inventory Management Software**

**1.Accounting**

**2.Inventory**

**3.Masters**

**4.Exit**

**Choose An Option : 2**

**1.Production 2.Closing Stock 3.Back**

**Enter Your choice : 1**

**1.Add 2.List 3.Back**

**Enter Your choice : 1**

**Query Executed Successfully!**

**Successfully Loaded Production Table!**

**Enter Date (DD-MM-YYYY) : 4-4-2023**

**Date:- 2023-04-04 00:00:00**

**Select Consumption item!**

**Selected item: Item1**

**Enter Quantity: 60**

**Enter Unit : KGS**

**Query Executed Successfully!**

**Query Executed Successfully!**

**Do You Want To ADD more Consumption Items?{y,n}: n**

**Select Generated item!**

**Selected item: Item2**

**Enter Quantity: 60**

**Enter Unit : PCS**

**Query Executed Successfully!**

**Query Executed Successfully!**

**Do You Want To ADD more Generated Items?{y,n}: n**

## Viewing Closing Stock

**Toh Kaise Hai APP Log!!!**

**Mini Accounting & Inventory Management Software**

**1.Accounting**

**2.Inventory**

**3.Masters**

**4.Exit**

**Choose An Option : 2**

**1.Production 2.Closing Stock 3.Back**

**Enter Your choice : 2**

**Selected item: Item1**

In_Date	PrtclrsIn	QntIn	UnitIn	PriceIn	AmountIn	Out_Date	PrtclrsOut	QntOut	UnitOut	PriceOut	AmountOut
2023-04-01	Purchase	200	KGS	55	11000	2023-04-01	Sales	15	KGS	100	1500
2023-04-06	Purchase	19	KGS	55	1045	2023-04-04	Production	60	KGS	1	1

**Closing Stock is : 144.0 UNITS**

## **Future Scope**

Future enhancements may include the integration of online transactions, mobile compatibility, and additional reporting features to further extend the software's capabilities.

## **Bibliography**

- [\*\*https://www.w3schools.com/\*\*](https://www.w3schools.com/)
- Sultan Chand's Computer Science with Python Textbook for class 12