DBMS PROJECT:(TAX SYSTEM)

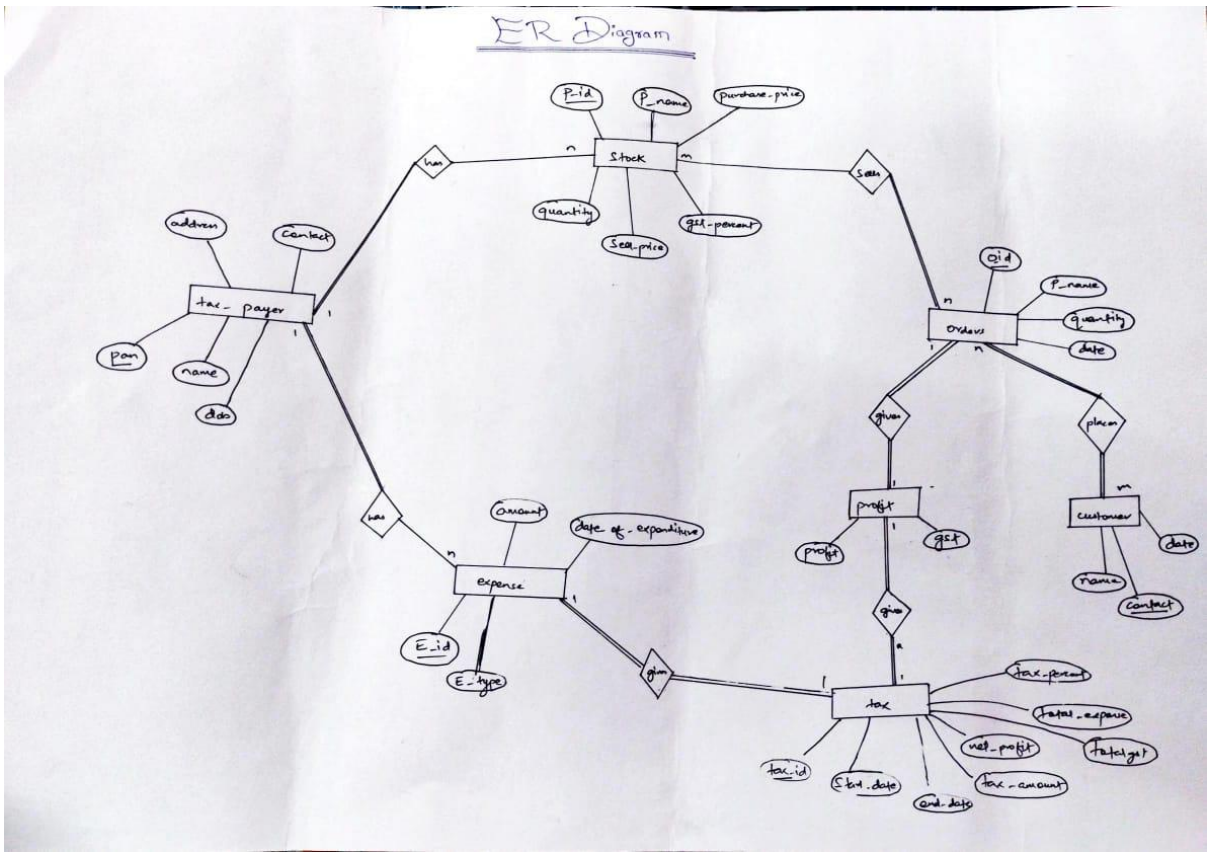
Name 1: Keshav Dalmia

SRN 1: PES1UG21CS275

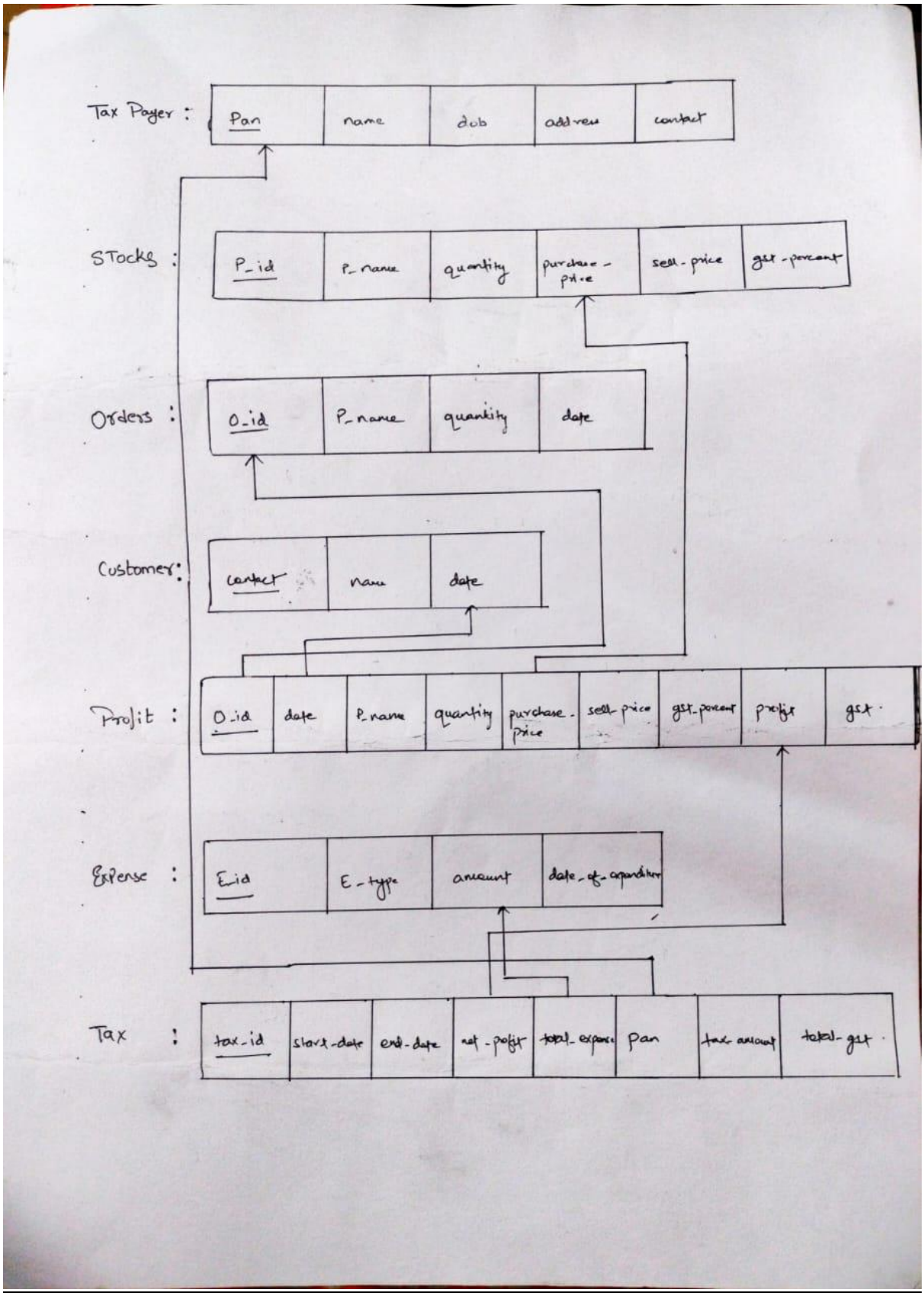
Name 2: Krishna Bhat

SRN 2: PES1UG21CS922

ER Diagram:



Relationship Schema:



CRUD Operations:

Tax_payer:

```python
def create_table_tax_payer_if_not_exists():
    mycursor = mydb.cursor()
    create_table_query = """
    CREATE TABLE IF NOT EXISTS tax_payer (
        pan varchar(10) NOT NULL PRIMARY KEY,
        name varchar(255) NOT NULL,
        dob date NOT NULL,
        address varchar(255) NOT NULL,
        contact varchar(10) NOT NULL
    )
    """
    mycursor.execute(create_table_query)

    print("Table created")
```

```
mysql> desc tax_payer;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| pan     | varchar(10)  | NO   | PRI | NULL    |       |
| name    | varchar(255) | NO   |     | NULL    |       |
| dob     | date         | NO   |     | NULL    |       |
| address | varchar(255) | NO   |     | NULL    |       |
| contact | varchar(10)  | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Table Stock:

```python
def create_table_stock_if_not_exists():
    mycursor = mydb.cursor()
    create_table_query = """
    CREATE TABLE IF NOT EXISTS stock (
        P_id varchar(10) NOT NULL PRIMARY KEY,
        P_name varchar(255) NOT NULL,
        quantity int DEFAULT NULL CHECK (quantity >= 0),
        purchase_price float NOT NULL,
        sell_price float NOT NULL,
        gst_percent int NOT NULL
    )
    """
    mycursor.execute(create_table_query)

    print("Table created")
```

```
mysql> desc stock;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| P_id           | varchar(10)  | NO   | PRI | NULL    |       |
| P_name         | varchar(255) | NO   |     | NULL    |       |
| quantity       | int          | YES  |     | NULL    |       |
| purchase_price | float        | NO   |     | NULL    |       |
| sell_price     | float        | NO   |     | NULL    |       |
| gst_percent    | int          | NO   |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Table Orders:

```python
def create_table_orders_if_not_exists():
    mycursor = mydb.cursor()
    create_table_query = """
    CREATE TABLE IF NOT EXISTS orders (
        O_id INT AUTO_INCREMENT PRIMARY KEY,
        P_name VARCHAR(225),
        quantity INT,
        date DATE
    )
    """



    # datequery = "UPDATE orders SET date = curDate();"

    mycursor.execute(create_table_query)
    # mycursor.execute(datequery)

    print("Table created")
```

```
mysql> desc orders;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| O_id     | int          | NO   | PRI | NULL    | auto_increment |
| P_name   | varchar(225) | YES  |     | NULL    |                |
| quantity | int          | YES  |     | NULL    |                |
| date     | date         | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Table Customer:

```python
def create_table_customer_if_not_exists():
    mycursor = mydb.cursor()
    create_table_query = """
    CREATE TABLE IF NOT EXISTS customer (
        name varchar(255) NOT NULL,
        contact varchar(10) NOT NULL PRIMARY KEY,
        date Date NOT NULL,
        frequency int NOT NULL
    )
    """
    mycursor.execute(create_table_query)

    print("Table created")
```

```
mysql> desc customer;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| name    | varchar(255) | NO   |     | NULL    |       |
| contact | varchar(10)  | NO   | PRI | NULL    |       |
| date    | date         | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Table Profit:

```python
def create_table_profit():
    mycursor = mydb.cursor()
    create_table_query = """
    CREATE TABLE IF NOT EXISTS profit (
        O_id INT AUTO_INCREMENT PRIMARY KEY,
        Date DATE,
        P_name VARCHAR(255),
        quantity INT,
        purchase_price float,
        sell_price float,
        gst_percent int,
        profit float,
        gst float
    )
    """
    mycursor.execute(create_table_query)

    print("Table 'profit' created")
```

```
mysql> desc profit;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| O_id           | int          | NO   | PRI | NULL    | auto_increment |
| Date           | date         | YES  |     | NULL    |                |
| P_name         | varchar(255) | YES  |     | NULL    |                |
| quantity       | int          | YES  |     | NULL    |                |
| purchase_price | float        | YES  |     | NULL    |                |
| sell_price     | float        | YES  |     | NULL    |                |
| gst_percent    | int          | YES  |     | NULL    |                |
| profit         | float        | YES  |     | NULL    |                |
| gst            | float        | YES  |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

Table Expense:

```python
def create_table_expense_if_not_exists():
    mycursor = mydb.cursor()
    create_table_query = """
    CREATE TABLE IF NOT EXISTS expense (
        E_id int AUTO_INCREMENT PRIMARY KEY,
        E_type varchar(255) NOT NULL,
        amount float NOT NULL,
        date_of_expenditure date NOT NULL
    )
    """
    mycursor.execute(create_table_query)

    print("Table created")
```

```
mysql> desc expense;
+---------------------+--------------+------+-----+---------+----------------+
| Field               | Type         | Null | Key | Default | Extra          |
+---------------------+--------------+------+-----+---------+----------------+
| E_id                | int          | NO   | PRI | NULL    | auto_increment |
| E_type              | varchar(255) | NO   |     | NULL    |                |
| amount              | float        | NO   |     | NULL    |                |
| date_of_expenditure | date         | NO   |     | NULL    |                |
+---------------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Table Tax:

```
def create_table_tax_if_not_exists():
    mycursor = mydb.cursor()
    create_table_query = """
    CREATE TABLE IF NOT EXISTS tax (
        pan varchar(10) NOT NULL,
        tax_id INT AUTO_INCREMENT PRIMARY KEY,
        start_date DATE NOT NULL,
        end_date DATE NOT NULL,
        net_profit FLOAT NOT NULL,
        total_expense FLOAT NOT NULL,
        tax_percentage FLOAT NOT NULL,D
        tax_amount FLOAT NOT NULL,
        total_gst FLOAT NOT NULL
    )
    """
    mycursor.execute(create_table_query)
    print("Table created Tax")
```

```
mysql> desc tax;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| pan            | varchar(10) | NO   |     | NULL    |                |
| tax_id         | int         | NO   | PRI | NULL    | auto_increment |
| start_date     | date        | NO   |     | NULL    |                |
| end_date       | date        | NO   |     | NULL    |                |
| net_profit     | float       | NO   |     | NULL    |                |
| total_expense  | float       | NO   |     | NULL    |                |
| tax_percentage | float       | NO   |     | NULL    |                |
| tax_amount     | float       | NO   |     | NULL    |                |
| total_gst      | float       | NO   |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

TRIGGERS USED:

1. To reduce the quantity in stock once the order is placed.

```
def create_trigger_reduce_quantity():
    mycursor = mydb.cursor()

    # Create trigger to reduce quantity from stock after an order is inserted
    trigger_query = """
    CREATE TRIGGER if not exists reduce_quantity_trigger
    AFTER INSERT ON orders
    FOR EACH ROW
    UPDATE stock
    SET quantity = quantity - NEW.quantity
    WHERE P_id = GetProductId (New.P_name);
    """
    mycursor.execute(trigger_query)

    print("Trigger created")
```

2. To update the profit once the order is completed..

```
def create_trigger_update_profit():
    mycursor = mydb.cursor()
    create_trigger_query = """
    CREATE TRIGGER if not exists update_profit
    AFTER INSERT ON orders
    FOR EACH ROW
    BEGIN
        INSERT INTO profit (Date, P_name, quantity, purchase_price, sell_price, gst_percent, profit, gst)
        SELECT
            NEW.date,
            NEW.P_name,
            NEW.quantity,
            stock.purchase_price,
            stock.sell_price,
            stock.gst_percent,
            (NEW.quantity * (stock.sell_price - stock.purchase_price)) AS profit,
            (NEW.quantity * stock.purchase_price * stock.gst_percent /100) as gst
        FROM
            stock
        WHERE
            stock.P_name = NEW.P_name;
    END;
    """
    mycursor.execute(create_trigger_query)

    print("Trigger 'update_profit' created")
```

3. To automatically update the date to the current date in various tables.

```
def date_trig():
    mycursor = mydb.cursor()
    date_query = """
        CREATE TRIGGER if not exists set_default_date
        BEFORE INSERT ON orders
        FOR EACH ROW
        SET NEW.date = IFNULL(NEW.date, CURDATE());
    """
    mycursor.execute(date_query)

    print("Trigger for date created")


def date_trig_Customer():
    mycursor = mydb.cursor()
    date_query = """
        CREATE TRIGGER if not exists set_default_date_customer
        BEFORE INSERT ON customer
        FOR EACH ROW
        SET NEW.date = IFNULL(NEW.date, CURDATE());
    """
    mycursor.execute(date_query)

    print("Trigger for date created")

def date_trig_for_expense():
    mycursor = mydb.cursor()
    date_query = """
        CREATE TRIGGER if not exists set_default_date_expense
        BEFORE INSERT ON expense
        FOR EACH ROW
        SET NEW.date_of_expenditure = IFNULL(NEW.date_of_expenditure, CURDATE());
    """
    mycursor.execute(date_query)
```

Procedures / Function:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `GetProductId`(p_name VARCHAR(255)) RETURNS
    READS SQL DATA
    DETERMINISTIC
BEGIN
    DECLARE product_id INT;

    SELECT stock.P_id INTO product_id
    FROM stock
    WHERE stock.P_name = p_name;

    RETURN product_id;
END
```

Entries In The Tables:

```
mysql> select * from tax_payer;
+------------+--------------+------------+--------------------+------------+
| pan        | name         | dob        | address            | contact    |
+------------+--------------+------------+--------------------+------------+
| ABCD123XYZ | KESHAV DALMIA | 2001-01-04 | JP NAGAR,BENGALURU | 8532172970 |
+------------+--------------+------------+--------------------+------------+
1 row in set (0.02 sec)
```

```
mysql> select * from stock;
+------+---------+----------+----------------+------------+-------------+
| P_id | P_name  | quantity | purchase_price | sell_price | gst_percent |
+------+---------+----------+----------------+------------+-------------+
| 1001 | Maggie  |     1000 |             10 |         14 |           5 |
| 1002 | Lays    |       51 |              8 |         10 |           5 |
| 1003 | Cake    |      100 |             15 |         20 |           6 |
| 1010 | Sprite  |       48 |             15 |         20 |           5 |
| 1111 | Kurkure |       75 |             15 |         20 |           5 |
+------+---------+----------+----------------+------------+-------------+
5 rows in set (0.02 sec)
```

```
mysql> select * from customer;
+--------------+------------+------------+
| name         | contact    | date       |
+--------------+------------+------------+
| John Doe     | 123456789  | 2023-11-23 |
| punit        | 1234567890 | 2023-11-23 |
| Kushang      | 2321712313 | 2023-11-23 |
| Krishna      | 8002931399 | 2023-11-22 |
| Keshav       | 8521772970 | 2023-11-22 |
| Dhruv        | 8521772971 | 2023-11-22 |
| Hriday       | 8521772975 | 2023-11-23 |
| Hriday       | 8521772976 | 2023-10-04 |
| Krishna Bhat | 8888777766 | 2023-11-23 |
+--------------+------------+------------+
9 rows in set (0.03 sec)
```

```
mysql> select * from tax;
+------------+--------+------------+------------+------------+--------------+----------------+------------+-----------+
| pan        | tax_id | start_date | end_date   | net_profit | total_expense | tax_percentage | tax_amount | total_gst |
+------------+--------+------------+------------+------------+--------------+----------------+------------+-----------+
| ABCD123XYZ |      1 | 2023-11-21 | 2023-11-23 |    20532.9 |          1200 |            0.1 |    1933.29 |   1933.29 |
+------------+--------+------------+------------+------------+--------------+----------------+------------+-----------+
1 row in set (0.00 sec)
```

```
mysql> select * from orders;
+------+-----------------+----------+------------+
| O_id | P_name          | quantity | date       |
+------+-----------------+----------+------------+
|   26 | Zim Zam         |       15 | 2023-11-21 |
|   45 | Maggie          |       10 | 2023-11-21 |
|   61 | Maggie          |       10 | 2023-11-22 |
|   62 | Zim Zam         |       20 | 2023-11-22 |
|   63 | britannia cake  |       10 | 2023-11-22 |
|   64 | Maggie          |       12 | 2023-11-22 |
|   65 | Zim Zam         |       16 | 2023-11-22 |
|   66 | Zim Zam         |       18 | 2023-11-22 |
|   67 | lays            |       25 | 2023-11-22 |
|   68 | coconut Rum     |        2 | 2023-11-22 |
|   69 | Coka Cola       |       10 | 2023-11-22 |
|   70 | maggie          |       10 | 2023-11-22 |
|   71 | lays            |      100 | 2023-11-22 |
|   72 | Maggie          |        8 | 2023-11-22 |
|   73 | coconut rum     |       40 | 2023-11-22 |
|   81 | mAGGIE          |       10 | 2023-11-23 |
|   82 | Maggie          |        2 | 2023-11-23 |
|   83 | Maggie          |        2 | 2023-11-23 |
|   84 | Maggie          |        2 | 2023-11-23 |
|   85 | Maggie          |        2 | 2023-11-23 |
|   86 | Kurkure         |        5 | 2023-11-23 |
|   87 | Maggie          |      182 | 2023-11-23 |
|   88 | Lays            |      949 | 2023-11-23 |
+------+-----------------+----------+------------+
23 rows in set (0.01 sec)
```

```
mysql> select * from expense;
+------+------------------+--------+----------------------+
| E_id | E_type           | amount | date_of_expenditure |
+------+------------------+--------+----------------------+
|    1 | Salary           |   1000 | 2023-11-22          |
|    2 | electricity bill |    200 | 2023-11-22          |
+------+------------------+--------+----------------------+
2 rows in set (0.02 sec)
```

```
mysql> select * from profit;
+------+------------+----------------+----------+----------------+------------+--------------+--------+--------+
| O_id | Date       | P_name         | quantity | purchase_price | sell_price | gst_percent | profit | gst    |
+------+------------+----------------+----------+----------------+------------+--------------+--------+--------+
|    1 | 2023-11-21 | Zim Zam        |       15 |            8.5 |         10 |            8 |   22.5 |  10.2 |
|    2 | 2023-11-21 | Maggie         |       10 |            9.5 |         12 |            5 |     25 |  4.75 |
|    3 | 2023-11-22 | Maggie         |       10 |            9.5 |         12 |            5 |     25 |  4.75 |
|    4 | 2023-11-22 | Zim Zam        |       20 |            8.5 |         10 |            8 |     30 |  13.6 |
|    5 | 2023-11-22 | britannia cake |       10 |            8.8 |         10 |            8 |     12 |  7.04 |
|    6 | 2023-11-22 | Maggie         |       12 |            9.5 |         12 |            5 |     30 |   5.7 |
|    7 | 2023-11-22 | Zim Zam        |       16 |            8.5 |         10 |            8 |     24 | 10.88 |
|    8 | 2023-11-22 | Zim Zam        |       18 |            8.5 |         10 |            8 |     27 | 12.24 |
|    9 | 2023-11-22 | lays           |       25 |             15 |         19 |           18 |    100 |  67.5 |
|   10 | 2023-11-22 | coconut Rum    |        2 |            950 |       1250 |           35 |    600 |   665 |
|   16 | 2023-11-22 | Coka Cola      |       10 |             14 |         19 |           18 |     59 |  25.2 |
|   17 | 2023-11-22 | maggie         |       10 |            9.5 |         12 |            5 |  26.25 |  4.75 |
|   18 | 2023-11-22 | lays           |      100 |             15 |         19 |           18 |    472 |   270 |
|   19 | 2023-11-22 | Maggie         |        8 |            9.5 |         12 |            5 |     21 |   3.8 |
|   20 | 2023-11-22 | coconut rum    |       40 |            950 |       1250 |           35 |  16200 | 13300 |
|   21 | 2023-11-23 | mAGGIE         |       10 |             10 |         14 |            5 |     42 |     5 |
|   22 | 2023-11-23 | Maggie         |        2 |             10 |         14 |            5 |    8.4 |     1 |
|   23 | 2023-11-23 | Maggie         |        2 |             10 |         14 |            5 |    8.4 |     1 |
|   24 | 2023-11-23 | Maggie         |        2 |             10 |         14 |            5 |    8.4 |     1 |
|   25 | 2023-11-23 | Maggie         |        2 |             10 |         14 |            5 |    8.4 |     1 |
|   26 | 2023-11-23 | Kurkure        |        5 |             15 |         20 |            5 |  26.25 |  3.75 |
|   27 | 2023-11-23 | Maggie         |      182 |             10 |         14 |            5 |  764.4 |    91 |
|   28 | 2023-11-23 | Lays           |      949 |              8 |         10 |            5 | 1992.9 | 379.6 |
+------+------------+----------------+----------+----------------+------------+--------------+--------+--------+
23 rows in set (0.00 sec)
```

Insert Operations:

```python
def insert_order(product_name, quant):
    mycursor = mydb.cursor()

    try:
        # SQL query to insert a record
        sql_query = "INSERT INTO orders (P_name, quantity) VALUES (%s, %s)"

        # Values to be inserted
        values = (product_name, quant)

        # Execute the query
        mycursor.execute(sql_query, values)

        # Commit the transaction
        mydb.commit()

        print("Record inserted successfully")

    except Exception as e:
        st.error(f"Error: {e}")
        # Rollback in case of an error
        mydb.rollback()

    finally:
        # Close the cursor and mydbection
        mycursor.close()
        mydb.close()
```

```python
def update_stock(product_data):
    mycursor = mydb.cursor()

    # Use INSERT ... ON DUPLICATE KEY UPDATE to handle insertion or update
    update_query = """
    INSERT INTO stock (P_id, P_name, quantity, purchase_price, sell_price, gst_percent)
    VALUES (%s, %s, %s, %s, %s, %s)
    ON DUPLICATE KEY UPDATE
    P_name = VALUES(P_name),
    quantity = quantity + VALUES(quantity),
    purchase_price = VALUES(purchase_price),
    sell_price = VALUES(sell_price),
    gst_percent = VALUES(gst_percent)
    """

    mycursor.executemany(update_query, product_data)

    mydb.commit()
    st.write("Stock Updated")
    print("Stock updated")

def insert_or_update_customer(name, contact):
```

```python
def insert_or_update_customer(name, contact):
    try:
        mycursor = mydb.cursor()

        # Insert customer, update contact and date if it already exists
        insert_query = """
            INSERT INTO customer (name, contact, date)
            VALUES (%s, %s, NOW())
            ON DUPLICATE KEY UPDATE contact = VALUES(contact), date =  NOW()
        """

        values = (name, contact)

        mycursor.execute(insert_query, values)
        mydb.commit()

        print("Record inserted or updated")
    except mysql.connector.Error as err:
        if err.errno == errorcode.ER_DUP_ENTRY:
            st.write(f"Duplicate entry for contact {contact}. Updating the existing record with the current date.")
            # Handle the update logic here
        else:
            print(f"Error: {err}")
```

```python
def insert_tax_information(pan,start_date, end_date, net_profit, total_expense, tax_percentage, tax_amount, total_gst):
    try:
        mycursor = mydb.cursor()

        # Insert tax information into the tax table
        insert_tax_query = """
            INSERT INTO tax (pan,start_date, end_date, net_profit, total_expense, tax_percentage, tax_amount, total_gst)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
        """
        tax_values = (pan,start_date, end_date, net_profit, total_expense, tax_percentage, tax_amount, total_gst)

        mycursor.execute(insert_tax_query, tax_values)
        mydb.commit()

        print("Tax information inserted into the tax table")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
```

User Aunthentication:

```python
# Function to authenticate users
def authenticate(username, password):
    if username == OWNER_USERNAME and password == OWNER_PASSWORD:
        return "owner"
    elif username == WORKER_USERNAME and password == WORKER_PASSWORD:
        return "worker"
    else:
        return None

# Function to check if the user is authenticated
def is_authenticated(user_role):
    return user_role is not None

# Function to render the login page
def render_login():
    st.title("Login")
    username = st.text_input("Username")
    password = st.text_input("Password", type="password")
    login_button = st.button("Login")

    if login_button:
        user_role = authenticate(username, password)
        if is_authenticated(user_role):
            st.session_state.user_role = user_role  # Store user_role in session state
            st.success(f"Login successful! Welcome, {user_role.capitalize()}!")
```