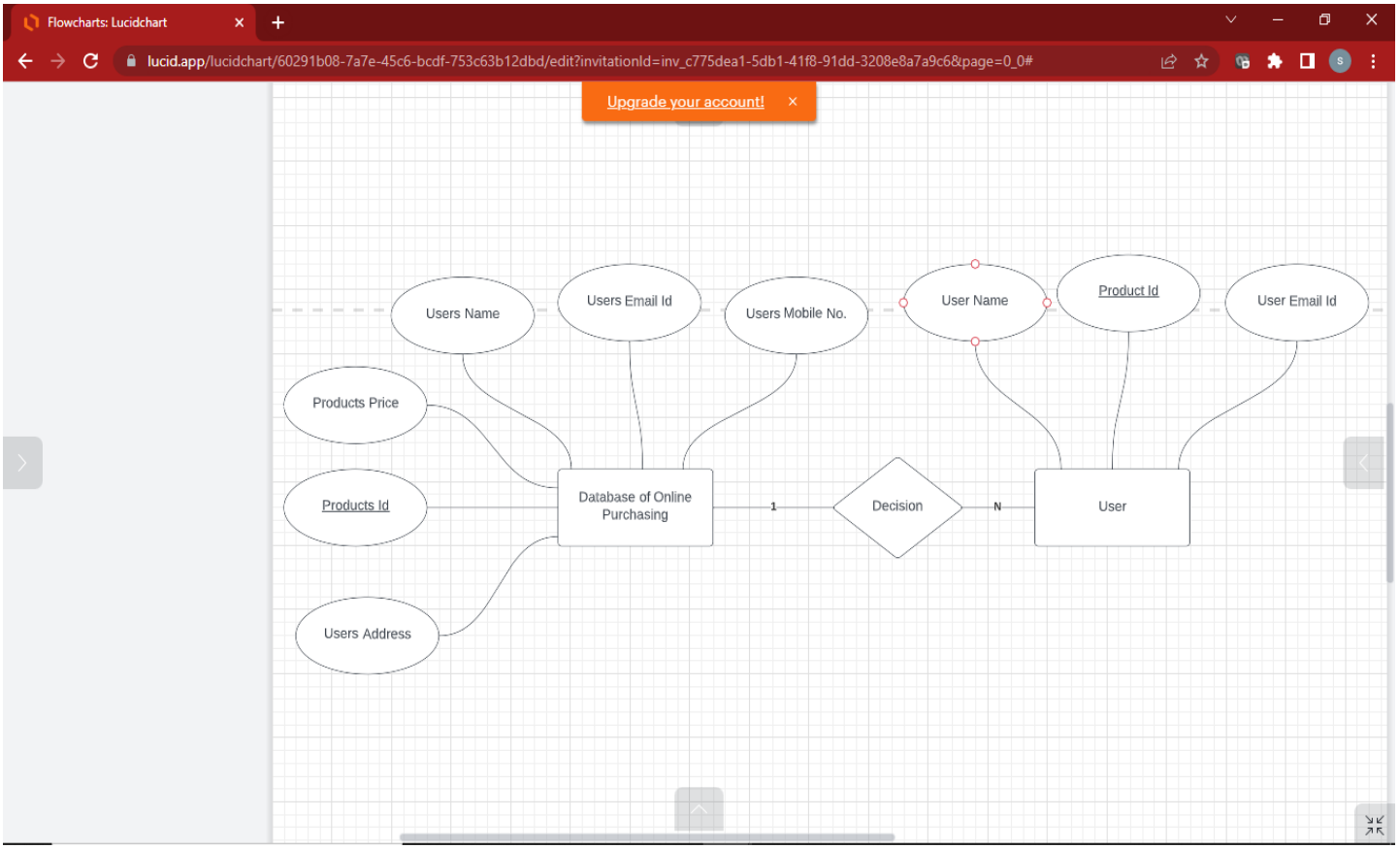
Ans – 1:- E R Diagram of purchasing databases



Ans-2 :- The following SQL scripts create the Product table in the selected database:

1. **CREATE TABLE** Product (
2. Product_id **INT PRIMARY KEY**,
3. Product_name **VARCHAR**(40),
4. Price **INT**,
5. Quantity **INT**
6.)

Next, execute the below scripts to insert data into this table:

1. **INSERT INTO** Product **VALUES**(111, 'Mobile', 10000, 10),
2. (112, 'Laptop', 20000, 15),
3. (113, 'Mouse', 300, 20),

4. (114, 'Hard Disk', 4000, 25),
5. (115, 'Speaker', 3000, 20);

Example of COMMIT Transaction

It's a good idea to divide the SQL Statements used in the transaction into multiple logical parts. And then, we can decide whether to commit or roll back the data. The following steps illustrate to create a transaction:

- Start the transaction using the **BEGIN TRANSACTION** command.
- Write the SQL statements and divide them based on our needs
- Use the **COMMIT** statement to complete the transaction and save the changes permanently.

Below are the commands that explain the COMMIT operations in SQL Server:

1. -- Start a new transaction
2. **BEGIN TRANSACTION**
3. -- SQL Statements
4. **INSERT INTO** Product **VALUES**(116, 'Headphone', 2000, 30)
5. **UPDATE** Product **SET** Price = 450 **WHERE** Product_id = 113
6. -- Commit changes
7. **COMMIT TRANSACTION**

Example of ROLLBACK Transaction

We will use the ROLLBACK command to undo any transactions that haven't been saved to the database yet and return to the point where the transaction began. **The following example explains the ROLLBACK operation in SQL Server:**

1. -- Start a new transaction
2. **BEGIN TRANSACTION**
3. -- SQL Statements
4. **UPDATE** Product **SET** Price = 5000 **WHERE** Product_id = 114
5. **DELETE FROM** Product **WHERE** Product_id = 116

Ans – 3:- An aggregate function in SQL performs a calculation on multiple values and returns a single value. SQL provides many aggregate functions that include avg, count, sum, min, max, etc. An aggregate function ignores NULL values when it performs the calculation, except for the count function.

An aggregate function in SQL returns one value after calculating multiple values of a column. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

Various types of SQL aggregate functions are:

- Count()
- Sum()
- Avg()
- Min()
- Max()

1:- COUNT() Function

The COUNT() function returns the number of rows in a database table.

2:- SUM() Function

The SUM() function returns the total sum of a numeric column.

3:- AVG() Function

The AVG() function calculates the average of a set of values.

4:- MIN() Function

The MIN() aggregate function returns the lowest value (minimum) in a set of non-NULL values.

5:- MAX() Function

The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.

Ans- 4:- You follow these steps to make a query a pivot table:

- First, select a base dataset for pivoting.
- Second, create a temporary result by using a derived table or common table expression (CTE)
- Third, apply the PIVOT operator.

```
SELECT * FROM
( SELECT
        category_name,
        product_id
    FROM
        production.products p
        INNER JOIN production.categories c
            ON c.category_id = p.category_id
) t
PIVOT(
    COUNT(product_id)
    FOR category_name IN (
        [Children Bicycles],
        [Comfort Bicycles],
        [Cruisers Bicycles],
        [Cyclocross Bicycles],
        [Electric Bikes],
        [Mountain Bikes],
        [Road Bikes])
) AS pivot_table;
```

Ans- 5:- SQL JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them**.**

Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table

- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table.

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

Ans-6 :- create table emp_salary(

empid int,

salary int

);

insert into emp_salary values
(100,20000),(101,25000),(102,22000),(103,30000),(104,26000),(105,32000)

select * from emp_salary order by salary desc limit 1 offset 3