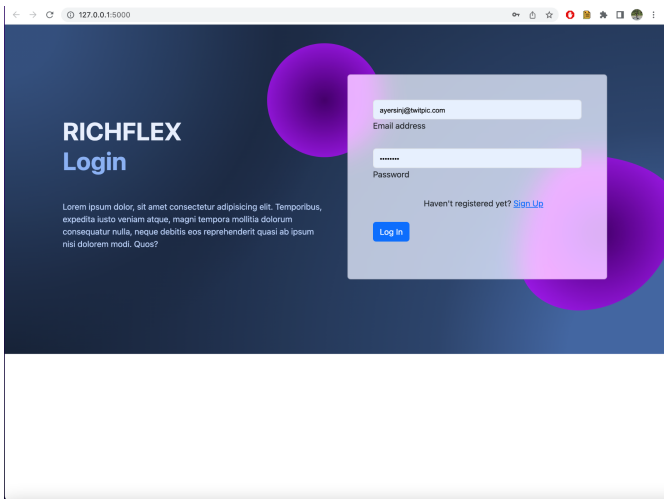
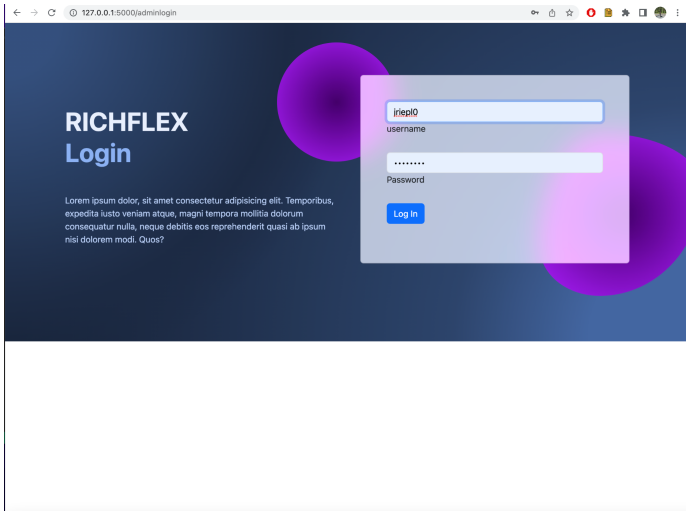
Project Deadline - 5

EMBEDDED QUERIES

1)

```
'SELECT * FROM customer where email = % s and password = % s', (c_id, password, )
```

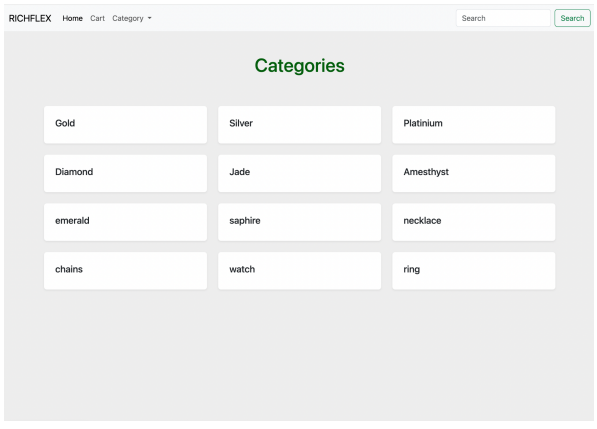


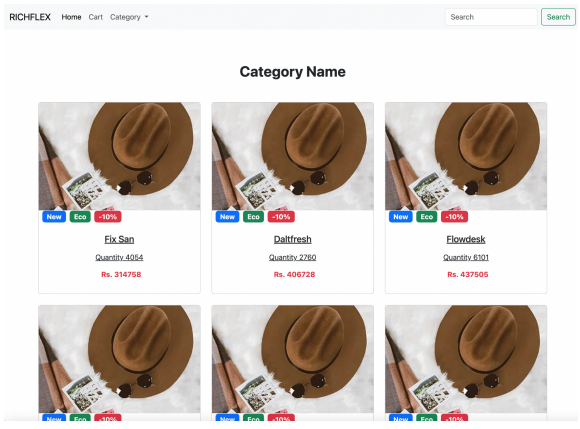



```
'SELECT * FROM Admin where username = % s and password = % s', (c_id, password, )
```

2)

```
"Select * from Category;
```





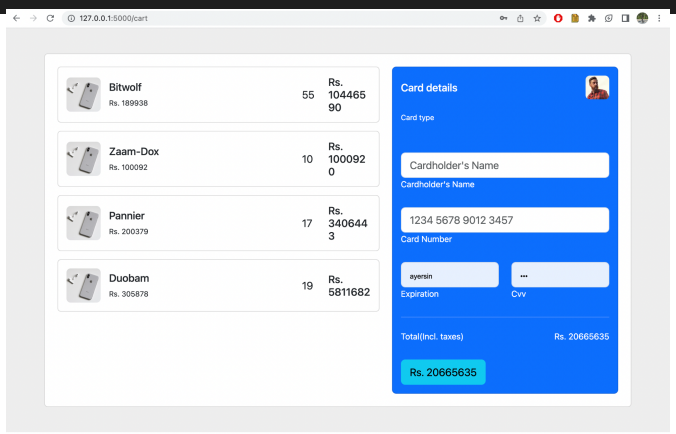
```
"select c.C_ID, p.nP_ID, p.PName, p.Price, c.Quantity, (p.Price * c.Quantity) AS
Total_Price FROM cart c,N_Product p where c.P_ID = p.nP_ID and c.C_ID =
%s",(session["c_id"],)
```

```
'select NP_id, Pname, quantity, price from N_Product n,Category c where
n.cat_id=c.cat_id and c.cat_id=% s',(id, );
```

3)

```
"select c.C_ID, SUM(p.Price * c.Quantity) AS Grand_Total FROM cart c,N_Product p where
c.P_ID = p.nP_ID and c.C_ID = %s GROUP BY c.C_ID",(session["c_id"],)


"select c.C_ID, SUM(p.Price * c.Quantity) AS Grand_Total FROM cart c,N_Product p where
c.P_ID = p.nP_ID and c.C_ID = %s GROUP BY c.C_ID",(session["c_id"],)
```




```
"select max(o_id) from orders"
```

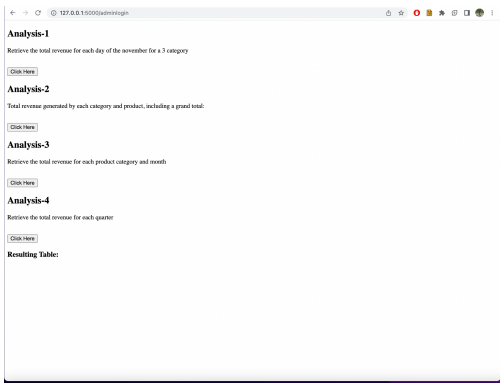
```
"INSERT INTO orders(O_ID, O_date, O_price, C_ID, P_ID, A_ID,PAY_ID) VALUES (%s,
CURRENT_DATE(), %s,%s,%s,2,0);",(m[0] +
1,data2[0]['grand_total'],session['c_id'],i['np_id'],)
```

OLAP QUERIES

1)

In this query, we have implemented a pivot OLAP Query,
Which retrieves the total revenue for each product category and month.
The admin department can use this data in analytics by providing insights into the sales performance of each product category for each month.

```
SELECT pc.Cat_ID,
       SUM(CASE WHEN MONTH(o.O_Date) = 1 THEN o.O_Price ELSE 0 END) AS
January_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 2 THEN o.O_Price ELSE 0 END) AS
February_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 3 THEN o.O_Price ELSE 0 END) AS March_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 4 THEN o.O_Price ELSE 0 END) AS April_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 5 THEN o.O_Price ELSE 0 END) AS May_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 6 THEN o.O_Price ELSE 0 END) AS June_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 7 THEN o.O_Price ELSE 0 END) AS July_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 8 THEN o.O_Price ELSE 0 END) AS August_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 9 THEN o.O_Price ELSE 0 END) AS
September_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 10 THEN o.O_Price ELSE 0 END) AS
October_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 11 THEN o.O_Price ELSE 0 END) AS
November_Revenue,
       SUM(CASE WHEN MONTH(o.O_Date) = 12 THEN o.O_Price ELSE 0 END) AS
December_Revenue
FROM ORDERS o, N_Product p ,Prod_Cat pc
where o.P_ID = p.NP_id
AND p.Cat_ID = pc.Cat_ID
GROUP BY pc.Cat_ID
order by pc.Cat_ID;
```

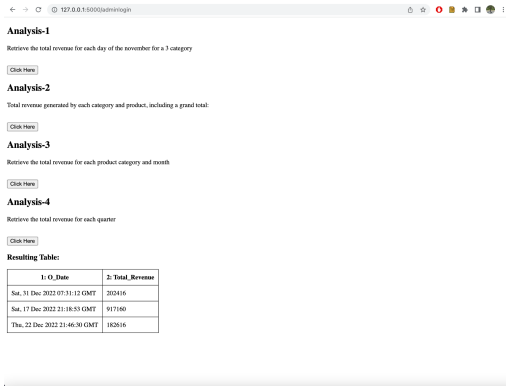


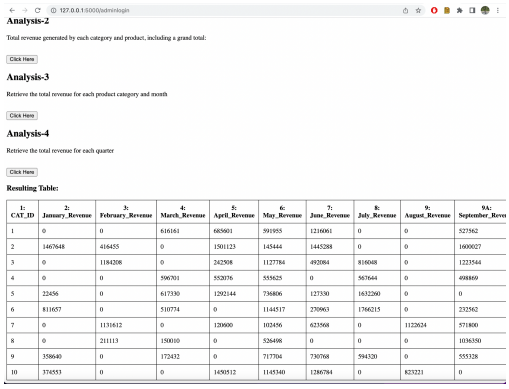
2)

In this query, we have implemented a Roll-up OLAP Query,
Which retrieves the total revenue for each quarter.

Using this query, businesses can easily track the revenue generated during each quarter and compare it with the previous years to analyze the business's performance over time.

```
SELECT YEAR(O_Date) AS Year, QUARTER(O_Date) AS Quarter, SUM(O_Price) AS
Total_Revenue
FROM ORDERS
GROUP BY YEAR(O_Date), QUARTER(O_Date) with rollup
order by YEAR(O_Date);
```

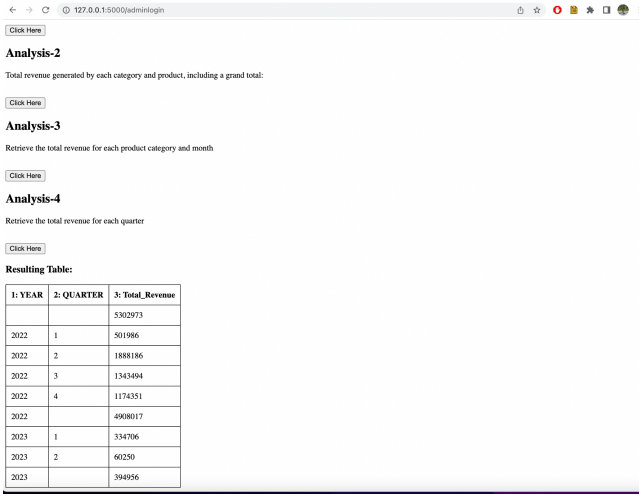




3)

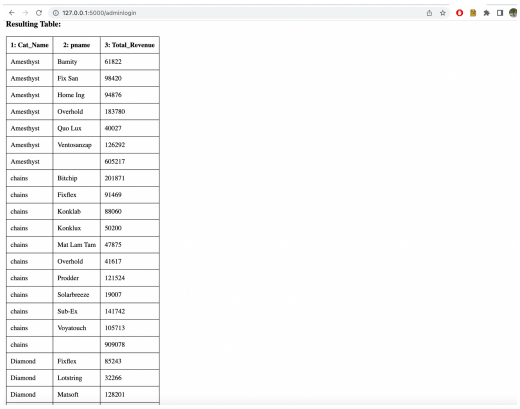
In this query, we have implemented a Drill-down OLAP Query,
Which retrieves the total revenue for each day of November for category 3.
This information can be useful for identifying areas of opportunity for marketing or inventory management, as well as tracking the effectiveness of promotional campaigns or changes in pricing strategies.

```
SELECT o.O_Date, SUM(o.O_Price) AS Total_Revenue
FROM ORDERS o,N_Product p , Prod_Cat pc
WHERE o.P_ID = p.NP_id
AND p.Cat_ID = pc.Cat_ID
AND YEAR(o.O_Date) = 2022
AND MONTH(o.O_Date) =12
AND pc.Cat_ID = 3
GROUP BY o.O_Date;
```



4) In this query, we have implemented a Roll-up OLAP Query,
Which retrieves the total revenue generated by each category and product, including a grand total.
Adding the WITH ROLLUP option allows the query to provide subtotal and total rows for each category and product, which can help identify trends and patterns in the data. This can be particularly useful for analyzing sales performance over time and identifying which products and categories drive the most revenue.

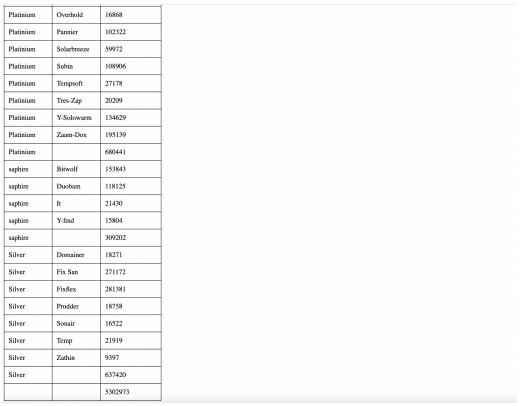
```
SELECT Cat_Name,pname, SUM(O_price) AS Total_Revenue
FROM Category, n_product, orders
where Category.Cat_ID = n_product.Cat_ID
and n_product.NP_ID = ORDERs.P_ID
GROUP BY Cat_Name,pname WITH ROLLUP;
```



127.0.0.1:5000/adminlogin

Resulting Table:

1: Cat_Name	2: pname	3: Total_Revenue
Amesthyst	Bamity	61822
Amesthyst	Fix San	98420
Amesthyst	Home Ing	94876
Amesthyst	Overhold	183780
Amesthyst	Quo Lux	40027
Amesthyst	Ventosanzap	126292
Amesthyst		605217
chains	Bitchip	201871
chains	Fixflex	91469
chains	Konklab	88060
chains	Konklux	50200
chains	Mat Lam Tam	47875
chains	Overhold	41617
chains	Prodder	121524
chains	Solarbreeze	19007
chains	Sub-Ex	141742
chains	Voyatouch	105713
chains		909078
Diamond	Fixflex	85243
Diamond	Lotstring	32266
Diamond	Matsoft	128201



Platinium	Overhold	16868
Platinium	Pannier	102322
Platinium	Solarbreeze	59972
Platinium	Subin	108906
Platinium	Tempsoft	27178
Platinium	Tres-Zap	20209
Platinium	Y-Solowarm	134629
Platinium	Zaam-Dox	195139
Platinium		680441
saphire	Bitwolf	153843
saphire	Duobam	118125
saphire	It	21430
saphire	Y-find	15804
saphire		309202
Silver	Domainer	18271
Silver	Fix San	271172
Silver	Fixflex	281381
Silver	Prodder	18758
Silver	Sonair	16522
Silver	Temp	21919
Silver	Zathin	9397
Silver		637420
		5302973

TRIGGERS

1) This trigger removes the items from the customer's cart after they have placed an order.

```
DELIMITER $$
CREATE TRIGGER remove_items_from_cart
AFTER INSERT ON ORDERS
FOR EACH ROW
BEGIN
```

```
  DELETE FROM Cart
  WHERE C_ID = NEW.C_ID AND P_ID = NEW.P_ID;
END
$$

-- to check the trigger, run these

INSERT INTO Cart (C_ID, P_ID, Quantity)
VALUES (1, 10, 2), (1, 20, 1);

INSERT INTO ORDERS(O_ID, O_date, O_price, C_ID, P_ID, A_ID,PAY_ID)
VALUES (102, '2023-04-01', 100, 1, 10, 1,101),
       (202, '2023-04-01', 50, 1, 20, 1,101);

-- CHECK IF THE CART IS EMPTY OR NOT

SELECT * FROM Cart WHERE C_ID = 1;
```

2)

This trigger reduces the number of products in the database after they have been added to a cart.

```
DELIMITER $$
CREATE TRIGGER reduce_product_quantity
AFTER INSERT ON cart
FOR EACH ROW
BEGIN
  UPDATE N_Product
  SET Quantity = Quantity - NEW.Quantity
  WHERE NP_ID = NEW.P_ID;
END $$

-- to check the trigger, run these
UPDATE N_product
SET Quantity = 1000
WHERE NP_ID = 10;


INSERT INTO Cart (C_ID, P_ID, Quantity)
VALUES (1, 1, 2);


-- Check if quantity has been reduced in the normal product table
SELECT * FROM N_Product WHERE NP_id = 1;
```