E- Commerce Management System

Rounak Bende, Parnavi Sen, Syona Jaimy

Abstract

In today's rapidly evolving business landscape, our e-commerce platform has witnessed substantial growth and expansion, making efficient data management and customer experience optimization paramount. To address these challenges and capitalize on opportunities, we propose the development of a comprehensive E-Commerce Database Management System.

Objectives

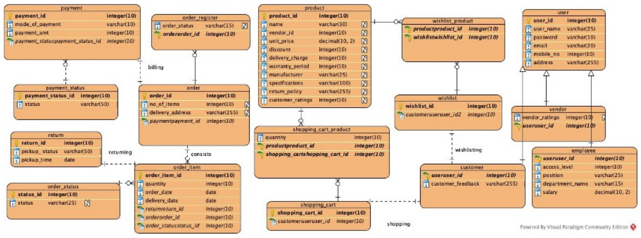
- Enhancing User Experience Improving Operational Efficiency
- Real-time Order Tracking
- Seamless Vendor Management
- Role-based Employee Access
- Efficient Returns Management
- Centralized Customer Feedback Hub

Introduction

The database has been meticulously crafted to serve as the backbone for a comprehensive e-commerce platform, addressing the diverse needs of an online retail environment. With the primary goal of streamlining operations, the system encompasses key functionalities such as product management, order processing, payment tracking, and user interactions. By facilitating efficient data storage and retrieval, the database aims to optimize the user experience, ensuring seamless transactions, timely order processing, and accurate inventory management. This project is driven by the ambition to create a robust foundation for an e-commerce application, fostering a user-friendly and reliable platform for both customers and administrators. The architecture of the database reflects careful consideration of design principles, including normalization techniques to eliminate data redundancy and maintain a high level of data integrity. The relational model incorporates foreign key constraints to establish relationships between entities, ensuring consistency in the representation of data across different tables. The subsequent sections delve deeper into the design decisions, data collection methods, and application features, providing a comprehensive overview of the database system.

Database Design

Entity Relationship Diagram



Database Schema

- Payment(payment_id, mode_of_payment, payment_amount, payment_status_id)
- Payment_Status(payment_status_id, status)
- Returns(return_id, pickup_status, pickup_time)
- Order_Register(order_id, order_status)
- Orders(order_id, no_of_items, delivery_address, payment_id)
- Order_Item(order_item_id, quantity, order_date, delivery_date,return_id, order_id, order_status_id)
- Order_Status(order_status_id, status)
- Product(product_id, name, vendor_id, unit_price, discount, delivery_charge, warranty_periodYears, manufacturer, specifications, return_policy, customer_ratings)
- User(user_id, user_name, password, email, mobile_no,address)
- Customer(user_id, feedback)
- Vendor(user_id, vendor_ratings)
- Employee(user_id, access_level, position, department_name, salary)
- Wishlist(wishlist_id, user_id)
- Wishlist_Product(wishlist_id, product_id)
- Shopping_Cart(shopping_cart_id, user_id)
- Shopping_Cart_Product(quantity, product_id, shopping_cart_id)

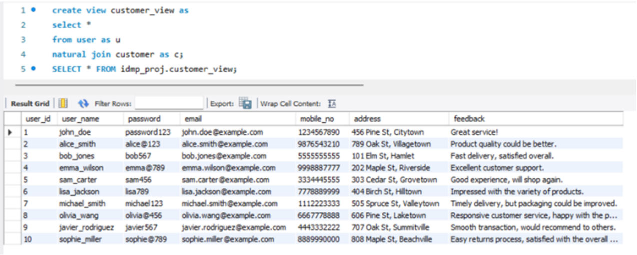
Implementation

Technologies Used

- Database Management System: MySQL
- Programming Language: Python
- Version Control: Git
- Python Driver: MySQL Connector/Python

Implementation Details

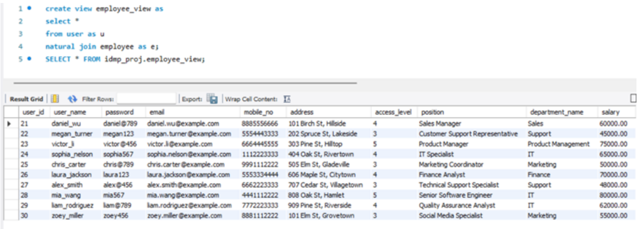
Views



```
create view customer_view as
select *
from user as u
natural join customer as c;
SELECT * FROM idmp_proj.customer_view;
```

user_id	user_name	password	email	mobile_no	address	feedback
1	john_doe	password123	john.doe@example.com	1234567890	456 Pine St, Citytown	Great service!
2	alice_smith	alice@123	alice.smith@example.com	9876543210	789 Oak St, Villagetown	Product quality could be better.
3	bob_jones	bob567	bob.jones@example.com	5555555555	101 Elm St, Hamlet	Fast delivery, satisfied overall.
4	emma_wilson	emma@789	emma.wilson@example.com	9998887777	202 Maple St, Riverside	Excellent customer support.
5	sam_carter	sam456	sam.carter@example.com	3334445555	303 Cedar St, Grovetown	Good experience, will shop again.
6	lisa_jackson	lisa789	lisa.jackson@example.com	7778889999	404 Birch St, Hilltown	Impressed with the variety of products.
7	michael_smith	michael123	michael.smith@example.com	1112223333	505 Spruce St, Valleytown	Timely delivery, but packaging could be improved.
8	olivia_wang	olivia@456	olivia.wang@example.com	6667778888	606 Pine St, Laketown	Responsive customer service, happy with the p...
9	javier_rodriguez	javier567	javier.rodriguez@example.com	4443332222	707 Oak St, Summitville	Smooth transaction, would recommend to others.
10	sophie_miller	sophie@789	sophie.miller@example.com	8889990000	808 Maple St, Beachville	Easy returns process, satisfied with the overall ...

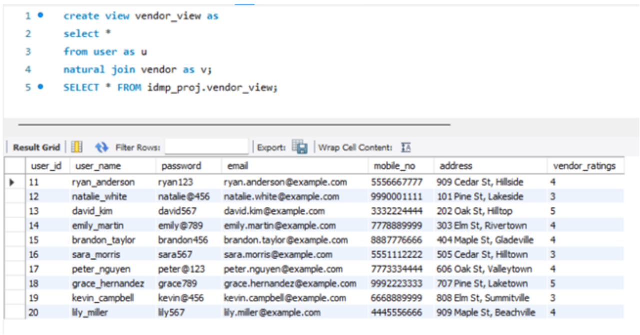
Figure 1: Customer view – It gives all the details of the customers



```
create view employee_view as
select *
from user as u
natural join employee as e;
SELECT * FROM idmp_proj.employee_view;
```

user_id	user_name	password	email	mobile_no	address	access_level	position	department_name	salary
21	daniel_wu	daniel@789	daniel.wu@example.com	8885556666	101 Birch St, Hillside	4	Sales Manager	Sales	60000.00
22	megan_turner	megan123	megan.turner@example.com	5554443333	202 Spruce St, Lakeside	3	Customer Support Representative	Support	45000.00
23	victor_li	victor@456	victor.li@example.com	6664445555	303 Pine St, Hilltop	5	Product Manager	Product Management	75000.00
24	sophia_nelson	sophia567	sophia.nelson@example.com	1112223333	404 Oak St, Rivertown	4	IT Specialist	IT	65000.00
25	chris_carter	chris@789	chris.carter@example.com	9991112222	505 Elm St, Gladeville	3	Marketing Coordinator	Marketing	50000.00
26	laura_jackson	laura123	laura.jackson@example.com	5553334444	606 Maple St, Citytown	4	Finance Analyst	Finance	70000.00
27	alex_smith	alex@456	alex.smith@example.com	6662223333	707 Cedar St, Villagetown	3	Technical Support Specialist	Support	48000.00
28	mia_wang	mia567	mia.wang@example.com	4441112222	808 Oak St, Hamlet	5	Senior Software Engineer	IT	80000.00
29	liam_rodriguez	liam@789	liam.rodriguez@example.com	7772223333	909 Pine St, Riverside	4	Quality Assurance Analyst	IT	62000.00
30	zoey_miller	zoey456	zoey.miller@example.com	8881112222	101 Elm St, Grovetown	3	Social Media Specialist	Marketing	55000.00

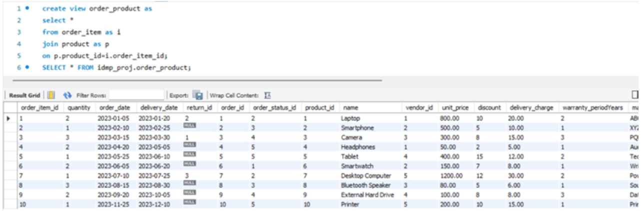
Figure 2: Employee view – It gives all the details of the employees



```
create view vendor_view as
select *
from user as u
natural join vendor as v;
SELECT * FROM idmp_proj.vendor_view;
```

user_id	user_name	password	email	mobile_no	address	vendor_ratings
11	ryan_anderson	ryan123	ryan.anderson@example.com	5556667777	909 Cedar St, Hillside	4
12	natalie_white	natalie@456	natalie.white@example.com	9990001111	101 Pine St, Lakeside	3
13	david_kim	david567	david.kim@example.com	3332224444	202 Oak St, Hilltop	5
14	emily_martin	emily@789	emily.martin@example.com	7778889999	303 Elm St, Rivertown	4
15	brandon_taylor	brandon456	brandon.taylor@example.com	8887776666	404 Maple St, Gladeville	4
16	sara_morris	sara567	sara.morris@example.com	5551112222	505 Cedar St, Hilltown	3
17	peter_nguyen	peter@123	peter.nguyen@example.com	7773334444	606 Oak St, Valleytown	4
18	grace_hernandez	grace789	grace.hernandez@example.com	9992223333	707 Pine St, Laketown	5
19	kevin_campbell	kevin@456	kevin.campbell@example.com	6668889999	808 Elm St, Summitville	3
20	lily_miller	lily567	lily.miller@example.com	4445556666	909 Maple St, Beachville	4

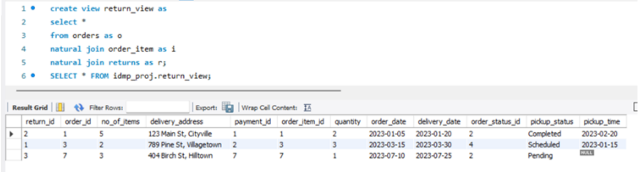
Figure 3: Vendor view – It gives all the details of the vendors



```
create view order_product as
select *
from order_item as i
join product as p
on p.product_id=i.order_item_id;
SELECT * FROM idmp_proj.order_product;
```

order_item_id	quantity	order_date	delivery_date	return_id	order_id	order_status_id	product_id	name	vendor_id	unit_price	discount	delivery_charge	warranty_periodYears	ma
1	2	2023-01-05	2023-01-20	2	1	2	1	Laptop	1	800.00	10	20.00	2	ABC
2	1	2023-02-10	2023-02-25	NULL	2	3	2	Smartphone	2	500.00	5	10.00	1	XYZ
3	3	2023-03-15	2023-03-30	1	3	4	3	Camera	3	300.00	8	15.00	3	PQ
4	2	2023-04-20	2023-05-05	NULL	4	5	4	Headphones	1	50.00	2	5.00	1	Au
5	1	2023-05-25	2023-06-10	NULL	5	5	5	Tablet	4	400.00	15	12.00	2	Tec
6	2	2023-06-05	2023-06-20	NULL	6	1	6	Smartwatch	2	150.00	7	8.00	1	Wri
7	1	2023-07-10	2023-07-25	3	7	2	7	Desktop Computer	5	1200.00	12	30.00	2	Pov
8	3	2023-08-15	2023-08-30	NULL	8	3	8	Bluetooth Speaker	3	80.00	5	6.00	1	Sou
9	2	2023-09-20	2023-10-05	NULL	9	4	9	External Hard Drive	4	100.00	8	8.00	3	Dat
10	1	2023-11-25	2023-12-10	NULL	10	5	10	Printer	5	200.00	10	15.00	1	Pri

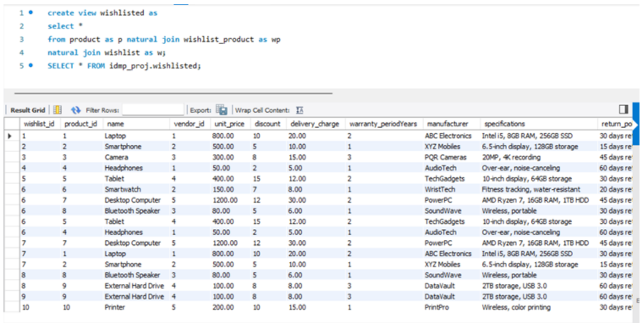
Figure 4: Order product view- It gives details of all order items in details.



```
create view return_view as
select *
from orders as o
natural join order_item as i
natural join returns as r;
SELECT * FROM idmp_proj.return_view;
```

return_id	order_id	no_of_items	delivery_address	payment_id	order_item_id	quantity	order_date	delivery_date	order_status_id	pickup_status	pickup_time
2	1	5	123 Main St, Cityville	1	1	2	2023-01-05	2023-01-20	2	Completed	2023-02-20
1	3	2	789 Pine St, Villagetown	2	3	3	2023-03-15	2023-03-30	4	Scheduled	2023-01-15
3	7	3	404 Birch St, Hilltown	7	7	1	2023-07-10	2023-07-25	2	Pending	NULL

Figure 5: Return view- It gives details of the order items that have return initiated

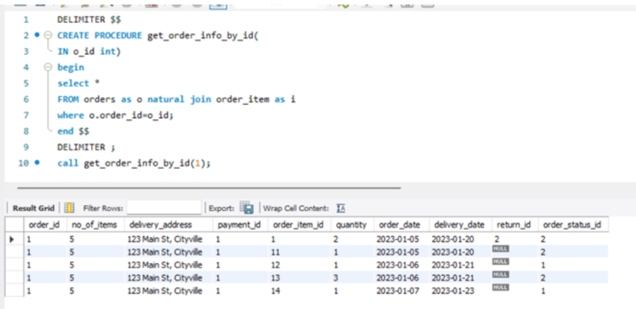


```
create view wishlisted as
select *
from product as p natural join wishlist_product as wp
natural join wishlist as w;
SELECT * FROM idmp_proj.wishlisted;
```

wishlist_id	product_id	name	vendor_id	unit_price	discount	delivery_charge	warranty_periodYears	manufacturer	specifications	return_po
1	1	Laptop	1	800.00	10	20.00	2	ABC Electronics	Intel i5, 8GB RAM, 256GB SSD	30 days re
2	2	Smartphone	2	500.00	5	10.00	1	XYZ Mobiles	6.5-inch display, 128GB storage	15 days re
3	3	Camera	3	300.00	8	15.00	3	PQR Cameras	20MP, 4K recording	45 days re
4	4	Headphones	1	50.00	2	5.00	1	AudioTech	Over-ear, noise-canceling	60 days re
5	5	Tablet	4	400.00	15	12.00	2	TechGadgets	10-inch display, 64GB storage	30 days re
6	6	Smartwatch	2	150.00	7	8.00	1	WristTech	Fitness tracking, water-resistant	20 days re
6	7	Desktop Computer	5	1200.00	12	30.00	2	PowerPC	AMD Ryzen 7, 16GB RAM, 1TB HDD	45 days re
6	8	Bluetooth Speaker	3	80.00	5	6.00	1	SoundWave	Wireless, portable	30 days re
6	5	Tablet	4	400.00	15	12.00	2	TechGadgets	10-inch display, 64GB storage	30 days re
6	4	Headphones	1	50.00	2	5.00	1	AudioTech	Over-ear, noise-canceling	60 days re
7	7	Desktop Computer	5	1200.00	12	30.00	2	PowerPC	AMD Ryzen 7, 16GB RAM, 1TB HDD	45 days re
7	1	Laptop	1	800.00	10	20.00	2	ABC Electronics	Intel i5, 8GB RAM, 256GB SSD	30 days re
7	2	Smartphone	2	500.00	5	10.00	1	XYZ Mobiles	6.5-inch display, 128GB storage	15 days re
8	8	Bluetooth Speaker	3	80.00	5	6.00	1	SoundWave	Wireless, portable	30 days re
8	9	External Hard Drive	4	100.00	8	8.00	3	DataVault	2TB storage, USB 3.0	60 days re
9	9	External Hard Drive	4	100.00	8	8.00	3	DataVault	2TB storage, USB 3.0	60 days re
10	10	Printer	5	200.00	10	15.00	1	PrintPro	Wireless, color printing	30 days re

Figure 6: Wishlisted view- It gives details of all the products that are wishlisted

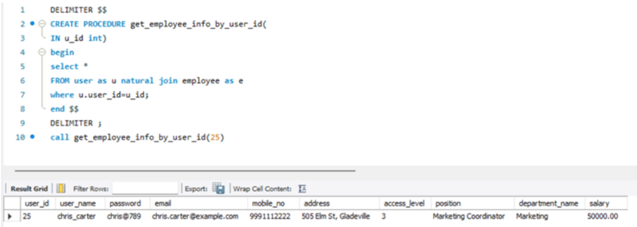
Stored Procedures



```sql
DELIMITER $$
CREATE PROCEDURE get_order_info_by_id(
IN o_id int)
begin
select *
FROM orders as o natural join order_item as i
where o.order_id=o_id;
end $$
DELIMITER ;
call get_order_info_by_id(1);
```

order_id	no_of_items	delivery_address	payment_id	order_item_id	quantity	order_date	delivery_date	return_id	order_status_id
1	5	123 Main St, Cityville	1	1	2	2023-01-05	2023-01-20	2	2
1	5	123 Main St, Cityville	1	11	1	2023-01-05	2023-01-20	NULL	2
1	5	123 Main St, Cityville	1	12	1	2023-01-06	2023-01-21	NULL	1
1	5	123 Main St, Cityville	1	13	3	2023-01-06	2023-01-21	NULL	2
1	5	123 Main St, Cityville	1	14	1	2023-01-07	2023-01-23	NULL	1

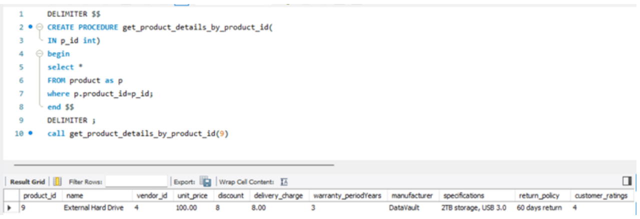
Figure 7: getorderinfobyid-Get all the order information when we provide order Id



```sql
DELIMITER $$
CREATE PROCEDURE get_employee_info_by_user_id(
IN u_id int)
begin
select *
FROM user as u natural join employee as e
where u.user_id=u_id;
end $$
DELIMITER ;
call get_employee_info_by_user_id(25)
```

user_id	user_name	password	email	mobile_no	address	access_level	position	department_name	salary
25	chris_carter	chris@789	chris.carter@example.com	9991112222	505 Elm St, Gladeville	3	Marketing Coordinator	Marketing	50000.00

Figure 8: getemployeeinfobyuserid- Get all employee info by using the user id

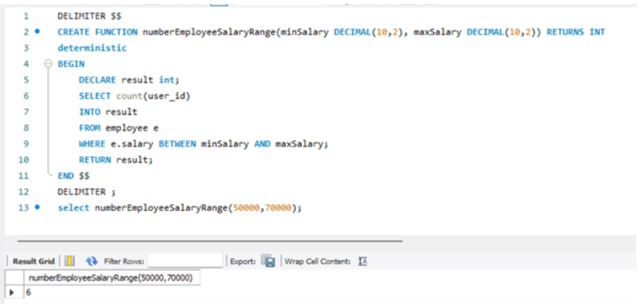


```sql
DELIMITER $$
CREATE PROCEDURE get_product_details_by_product_id(
IN p_id int)
begin
select *
FROM product as p
where p.product_id=p_id;
end $$
DELIMITER ;
call get_product_details_by_product_id(9)
```

product_id	name	vendor_id	unit_price	discount	delivery_charge	warranty_periodYears	manufacturer	specifications	return_policy	customer_ratings
9	External Hard Drive	4	100.00	8	8.00	3	DataVault	2TB storage, USB 3.0	60 days return	4

Figure 9: getproductdetailsbyproductid- Get all product details using the product id

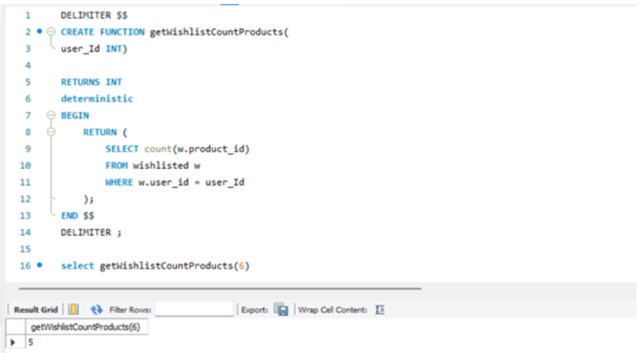
Function



```sql
DELIMITER $$
CREATE FUNCTION numberEmployeeSalaryRange(minSalary DECIMAL(10,2), maxSalary DECIMAL(10,2)) RETURNS INT
deterministic
BEGIN
    DECLARE result int;
    SELECT count(user_id)
    INTO result
    FROM employee e
    WHERE e.salary BETWEEN minSalary AND maxSalary;
    RETURN result;
END $$
DELIMITER ;
select numberEmployeeSalaryRange(50000,70000);
```

numberEmployeeSalaryRange(50000,70000)
6

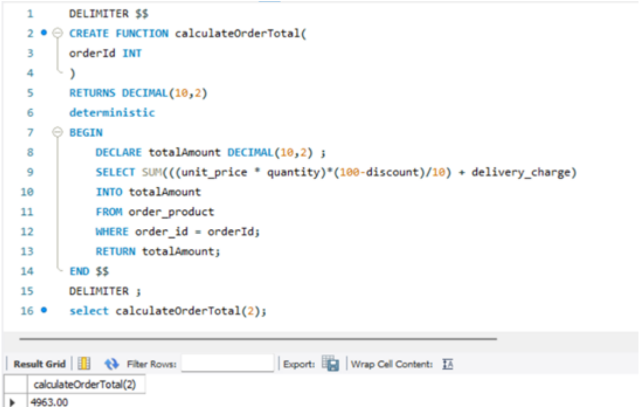
Figure 10: numberEmployeeSalaryRange- This function takes the minimum salary and maximum salary value and returns the number of employees that have salary in that range



```sql
DELIMITER $$
CREATE FUNCTION getWishlistCountProducts(
user_Id INT)

RETURNS INT
deterministic
BEGIN
    RETURN (
        SELECT count(w.product_id)
        FROM wishlisted w
        WHERE w.user_id = user_Id
    );
END $$
DELIMITER ;

select getWishlistCountProducts(6)
```

getWishlistCountProducts(6)
5

Figure 11: calculateOrderTotal- This function takes the order Id and returns the subtotal of all the order items in that order



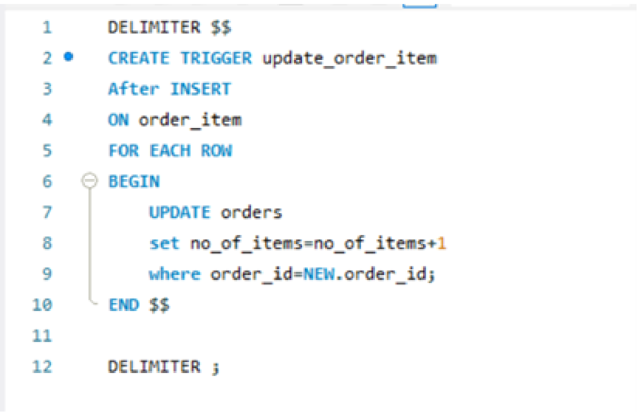
```sql
DELIMITER $$
CREATE FUNCTION calculateOrderTotal(
orderId INT
)
RETURNS DECIMAL(10,2)
deterministic
BEGIN
    DECLARE totalAmount DECIMAL(10,2) ;
    SELECT SUM(((unit_price * quantity)*(100-discount)/10) + delivery_charge)
    INTO totalAmount
    FROM order_product
    WHERE order_id = orderId;
    RETURN totalAmount;
END $$
DELIMITER ;
select calculateOrderTotal(2);
```

calculateOrderTotal(2)
4963.00

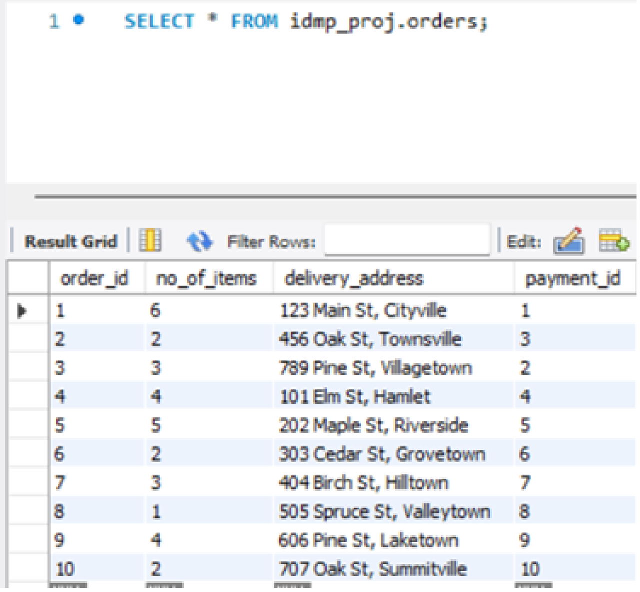
Figure 12: getWishlistCountProducts-This function takes in the user Id and returns the number of products in the wishlist

Trigger

UpdateOrderItem: increments the total number of items in the associated order table whenever a new item is inserted into the orderitem table.



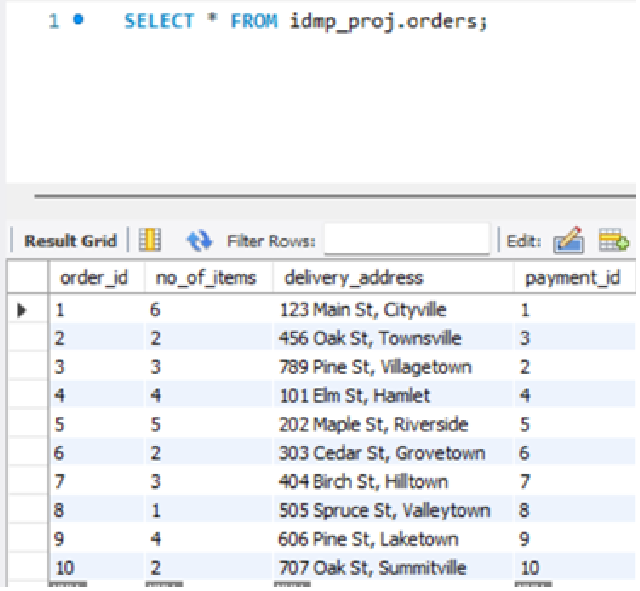
```
DELIMITER $$
CREATE TRIGGER update_order_item
After INSERT
ON order_item
FOR EACH ROW
BEGIN
    UPDATE orders
    set no_of_items=no_of_items+1
    where order_id=NEW.order_id;
END $$

DELIMITER ;
```



```
SELECT * FROM idmp_proj.orders;
```

order_id	no_of_items	delivery_address	payment_id
1	6	123 Main St, Cityville	1
2	2	456 Oak St, Townsville	3
3	3	789 Pine St, Villagetown	2
4	4	101 Elm St, Hamlet	4
5	5	202 Maple St, Riverside	5
6	2	303 Cedar St, Grovetown	6
7	3	404 Birch St, Hilltown	7
8	1	505 Spruce St, Valleytown	8
9	4	606 Pine St, Laketown	9
10	2	707 Oak St, Summitville	10

Figure 13: Before insertion



```
SELECT * FROM idmp_proj.orders;
```

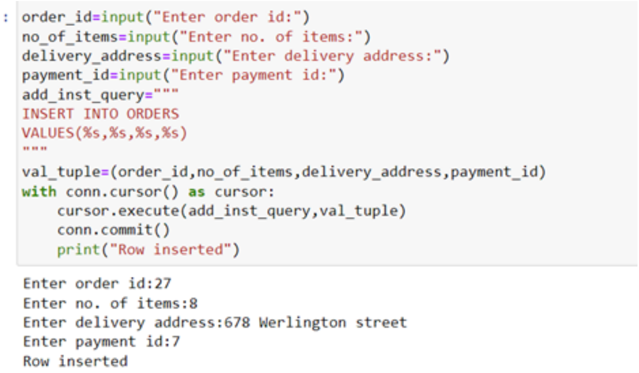
order_id	no_of_items	delivery_address	payment_id
1	6	123 Main St, Cityville	1
2	2	456 Oak St, Townsville	3
3	3	789 Pine St, Villagetown	2
4	4	101 Elm St, Hamlet	4
5	5	202 Maple St, Riverside	5
6	2	303 Cedar St, Grovetown	6
7	3	404 Birch St, Hilltown	7
8	1	505 Spruce St, Valleytown	8
9	4	606 Pine St, Laketown	9
10	2	707 Oak St, Summitville	10

Figure 14: After Insert orderitem for orderid 3

Python Database Connection Interface:

Used mysql connector to connect the python file to the database.

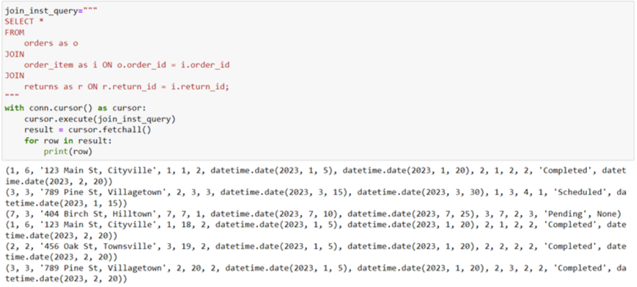
- The python code connects to the database and allows us to insert, update and delete rows.



```
order_id=input("Enter order id:")
no_of_items=input("Enter no. of items:")
delivery_address=input("Enter delivery address:")
payment_id=input("Enter payment id:")
add_inst_query="""
INSERT INTO ORDERS
VALUES(%s,%s,%s,%s)
"""
val_tuple=(order_id,no_of_items,delivery_address,payment_id)
with conn.cursor() as cursor:
    cursor.execute(add_inst_query,val_tuple)
    conn.commit()
    print("Row inserted")
```

```
Enter order id:27
Enter no. of items:8
Enter delivery address:678 Werlington street
Enter payment id:7
Row inserted
```

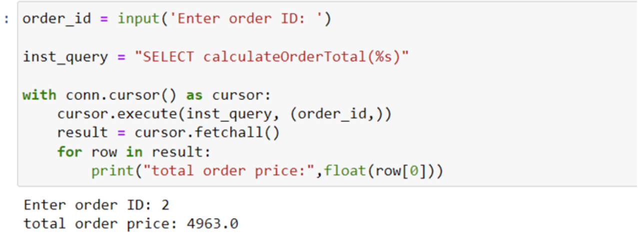
- It can join 3 tables together. The python code joins 3 tables and gives us the information of all the order items that need to be returned.



```
join_inst_query="""
SELECT *
FROM
    orders as o
JOIN
    order_item as i ON o.order_id = i.order_id
JOIN
    returns as r ON r.return_id = i.return_id;
"""
with conn.cursor() as cursor:
    cursor.execute(join_inst_query)
    result = cursor.fetchall()
    for row in result:
        print(row)
```

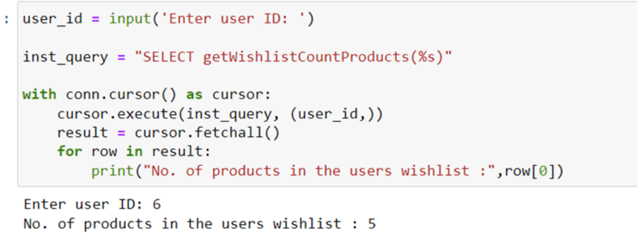
```
(1, 6, '123 Main St, Cityville', 1, 1, 2, datetime.date(2023, 1, 5), datetime.date(2023, 1, 20), 2, 1, 2, 2, 'Completed', datet
ime.date(2023, 2, 20))
(3, 3, '789 Pine St, Villagetown', 2, 3, 3, datetime.date(2023, 3, 15), datetime.date(2023, 3, 30), 1, 3, 4, 1, 'Scheduled', da
tetime.date(2023, 1, 15))
(7, 3, '404 Birch St, Hilltown', 7, 7, 1, datetime.date(2023, 7, 10), datetime.date(2023, 7, 25), 3, 7, 2, 3, 'Pending', None)
(1, 6, '123 Main St, Cityville', 1, 18, 2, datetime.date(2023, 1, 5), datetime.date(2023, 1, 20), 2, 1, 2, 2, 'Completed', date
time.date(2023, 2, 20))
(2, 2, '456 Oak St, Townsville', 3, 19, 2, datetime.date(2023, 1, 5), datetime.date(2023, 1, 20), 2, 2, 2, 2, 'Completed', date
time.date(2023, 2, 20))
(3, 3, '789 Pine St, Villagetown', 2, 20, 2, datetime.date(2023, 1, 5), datetime.date(2023, 1, 20), 2, 3, 2, 2, 'Completed', da
tetime.date(2023, 2, 20))
```

- It can also get information from the user and run the different functions.



```
order_id = input('Enter order ID: ')

inst_query = "SELECT calculateOrderTotal(%s)"

with conn.cursor() as cursor:
    cursor.execute(inst_query, (order_id,))
    result = cursor.fetchall()
    for row in result:
        print("total order price:",float(row[0]))
```

```
Enter order ID: 2
total order price: 4963.0
```



```
user_id = input('Enter user ID: ')

inst_query = "SELECT getWishlistCountProducts(%s)"

with conn.cursor() as cursor:
    cursor.execute(inst_query, (user_id,))
    result = cursor.fetchall()
    for row in result:
        print("No. of products in the users wishlist :",row[0])
```

```
Enter user ID: 6
No. of products in the users wishlist : 5
```

Database Insights

The following graphs shows some of the distributions of data.

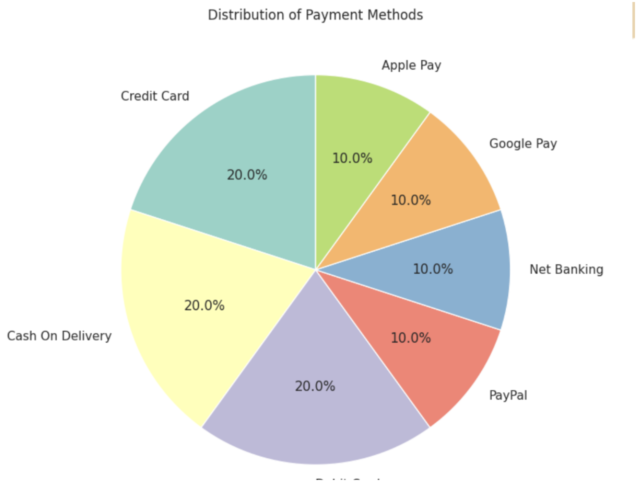


Figure 15: the Distribution of Payment Methods

the most popular payment methods were Cash on delivery, Debit card and Credit card.

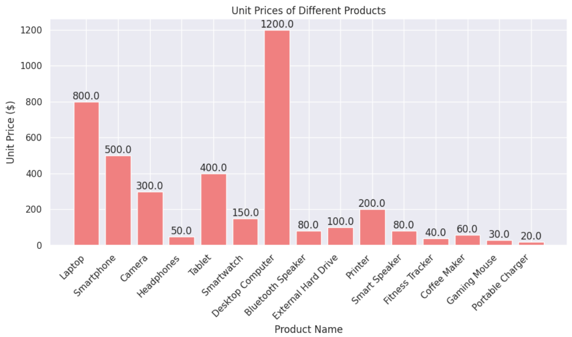


Figure 16: Figure shows the product and their prices

Computer Desktop is the most expensive item being sold on this platform whereas portable charger is the least expensive.

Future Scope

1. **Enhanced User Personalization:**

   Future iterations of the system could implement personalized recommendations based on user preferences and shopping history, enhancing the overall user experience and encouraging customer engagement.

2. **Expanded Vendor Analytics:**

   Developing advanced analytics tools for vendors would provide deeper insights into product performance, allowing vendors to make informed decisions and optimize their product offerings. This could include features such as predictive analytics and trend analysis.

3. **Global Scalability:**

   Designing the system architecture to accommodate international expansion and increased user volume is crucial for future growth. Considerations for multi-language support, currency conversion, and regional regulations should be incorporated to ensure a seamless global user experience.

4. **Scalability:**

   Ensuring that the system architecture is designed to accommodate future growth and increased demands on the system is paramount. This involves scalability in terms of both user base and product catalog expansion. Scalability measures should be implemented to handle higher transaction volumes efficiently.

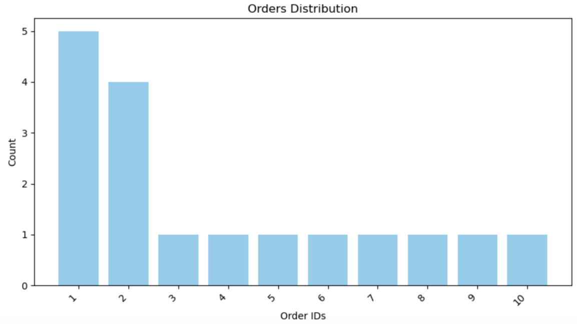


Figure 17: Figure shows the count of items in a single order

Here we see that order ID=1 has most number of items

Conclusion

In summary, our e-commerce database design prioritizes data integrity, security, and scalability. The user, product, and order tables, carefully normalized, form a solid foundation. Robust encryption safeguards sensitive data, while indexing and optimization enhance query performance. The system is designed with scalability in mind, providing a responsive platform for current and future needs. Implementation of these measures ensures a reliable, secure, and scalable e-commerce solution.