EAST WEST UNIVERSITY

Wholesale Management System

Course Name: **Database Systems**

Course: **CSE302**

Section: **04**

Group No: **11**

Submitted By:

Members Name	**ID**	**Contribution**
MD. Fahim	_____-____-_____	33.33%
Md. Adnan Morshed	_____-____-_____	33.33%
Md. Ahnaf Morshed	_____-____-_____	33.33%

Submitted To

Md Mostofa Kamal Rasel

Assistant Professor,

Department of Computer Science and Engineering.

Submitted on**:** --12th January 2022

Wholesale Management System

Project Description: -

In our wholesale management system, we can maintain the details of the products like their id, category, name, and stock status. We can also maintain the details of the customers like their name, email address, customer id, contact number, and so on. We have the order id to see the orders and have the payment id to see which orders have their payment. We can see the list of the customers who have done the payment. We can also generate monthly sales as well.

Objective: -

We wanted to create an efficient model with data redundancy prevention. It means there will be no data redundancy in our management system. The management system is cost-effective and can be easily accessed and maintained. Furthermore, we have also kept in mind the functional dependencies while designing the database. We can update or modify the system as well. We can use the wholesale management system to maintain the entire buying and selling procedures and keep all its records.

Entity sets:

All entity sets of this database are strong entity sets except the delivery entity set. Because each entity set contains a primary key of its own but delivery is dependent on the product and orders entity sets.

Product:

The product keeps the information about the product id, product name, category, and product description. It also has the unit price which is the price of buying products and the list price which is the price of selling products. It also contains the stock status and quantity. If the quantity is less than 4 then it will be called out of stock.

Product
product_Id Product_Name Category List_price Product_description Unit_price Stock_Status Product_Quantity

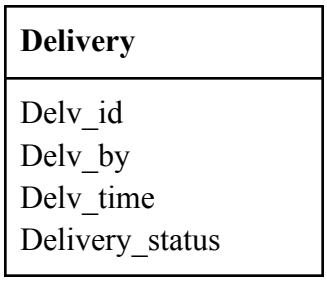
Orders:

Orders entity set keeps the information about the order id, Order_Date payment_status, quantity. Here, Order_Date is to record the date of the orders and quantity stores the quantity of a product that has been ordered, and payment_satus shows whether the order is paid or not. It also has a derived attribute named payable, which will be derived from the multiplication of quantity and list price from the entity set product. Hence payable shows the amount of the total price of the order.

Orders
Order_id Order_Date payment_status Quantity payable( )

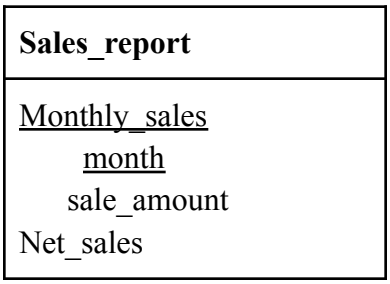
Delivery:

Delivery keeps the information about the delivery id, the person who will deliver the order, delivery time, and delivery status. Here delivery status will show whether the order has been delivered or not. This is a weak entity set. It is dependent on the product and orders entity sets.



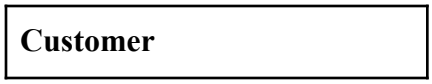
Sales_report:

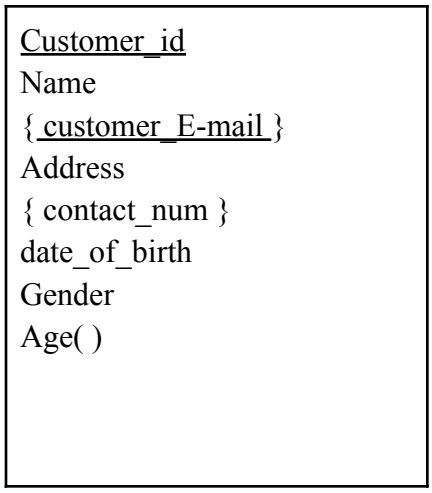
Sales report keeps the information about monthly sales. Monthly sale is a composite attribute that has month and sale amount. It keeps the record of the total sales of each month. The entity also contains Net sales. Net sales keep the record of total sales from the beginning till now.



Customer:

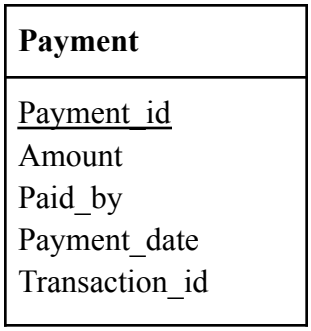
The customer keeps the information about the customer id, name, address, date of birth, and gender. It also has the customer's email address and contact number which are multivalued. It means a customer can have more than one email id or contact number. It also has a derived attribute named age which will be derived from the birth date of the customer.





Payment:

Payment keeps the information about the payment id , amount and payment date and the customer name who has done the payment. It also contains the transaction id as well which confirms the payment.



Cardinality constraints and participation:

Customer– Orders:

Cardinality constraint: The cardinality constraint from the "customer" entity set to the "orders" entity set is one–to–many. This means a customer may have many orders and orders may have only one customer.

Participation: Customer is mandatory for orders and orders are optional for customers.

Orders – product:

Cardinality constraint: The "orders" entity set to the "product" entity set is many–to–many. This means an order may have more than one gallery, as well as a product, and may have more than one artist.

Participation: Product is mandatory for orders but orders are optional for the product.

Orders – Payment:

Cardinality constraint: The "Orders" entity set to the "Payment" entity set is one–to–one. This order may have only one payment and payment has also only one order.

Participation: Order is mandatory for payment but payment is optional for order.

Payment – Sales_report:

Cardinality constraint: The "Orders" entity set to the "Payment" entity set is one–to–one. These orders may have only one payment and payment has also only one order.

Participation: Both are optional for each other.

Orders – Delivery:

Cardinality constraint: The "Orders" entity set to the "Delivery" entity set is many–to–one. This delivery entity may have many orders but the order entity has one delivery.

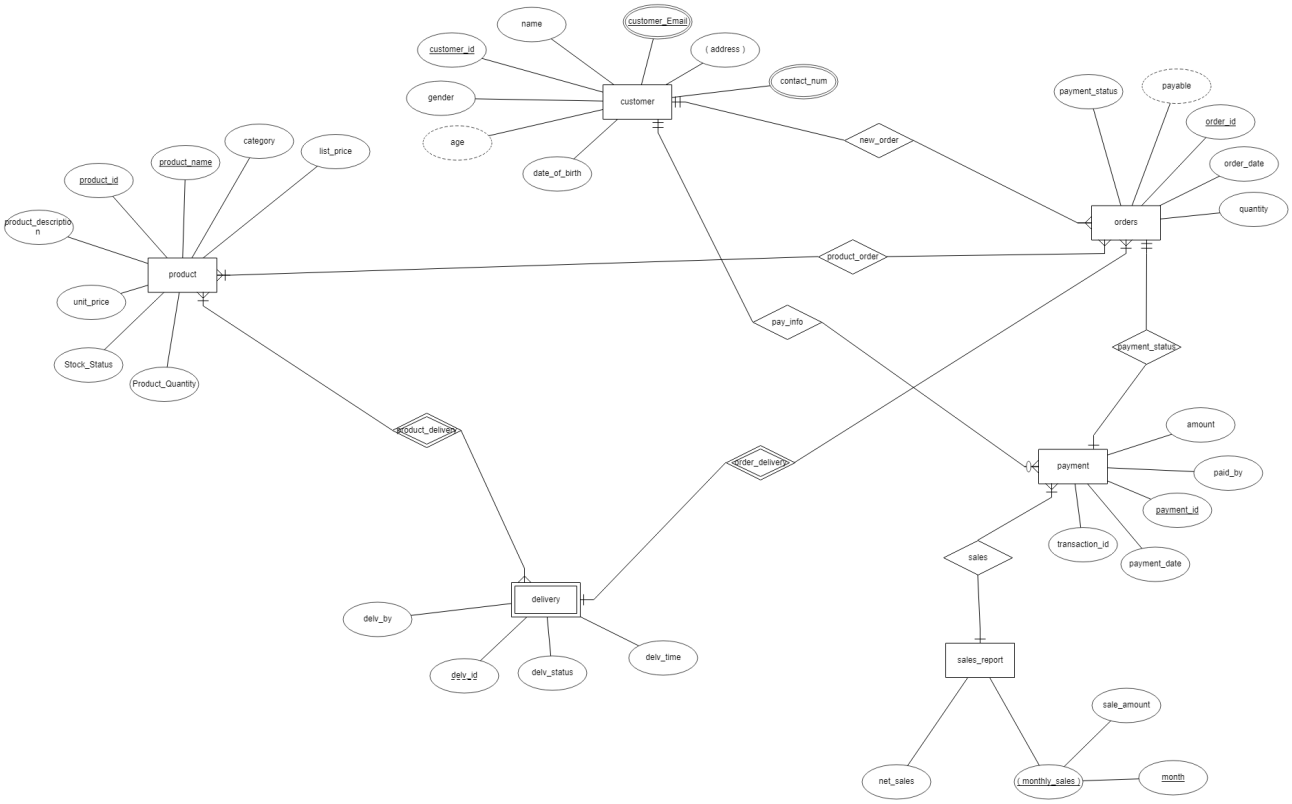
Participation: The ordering entity is mandatory for delivery but the delivery entity is optional for the order entity.

Product – Delivery:

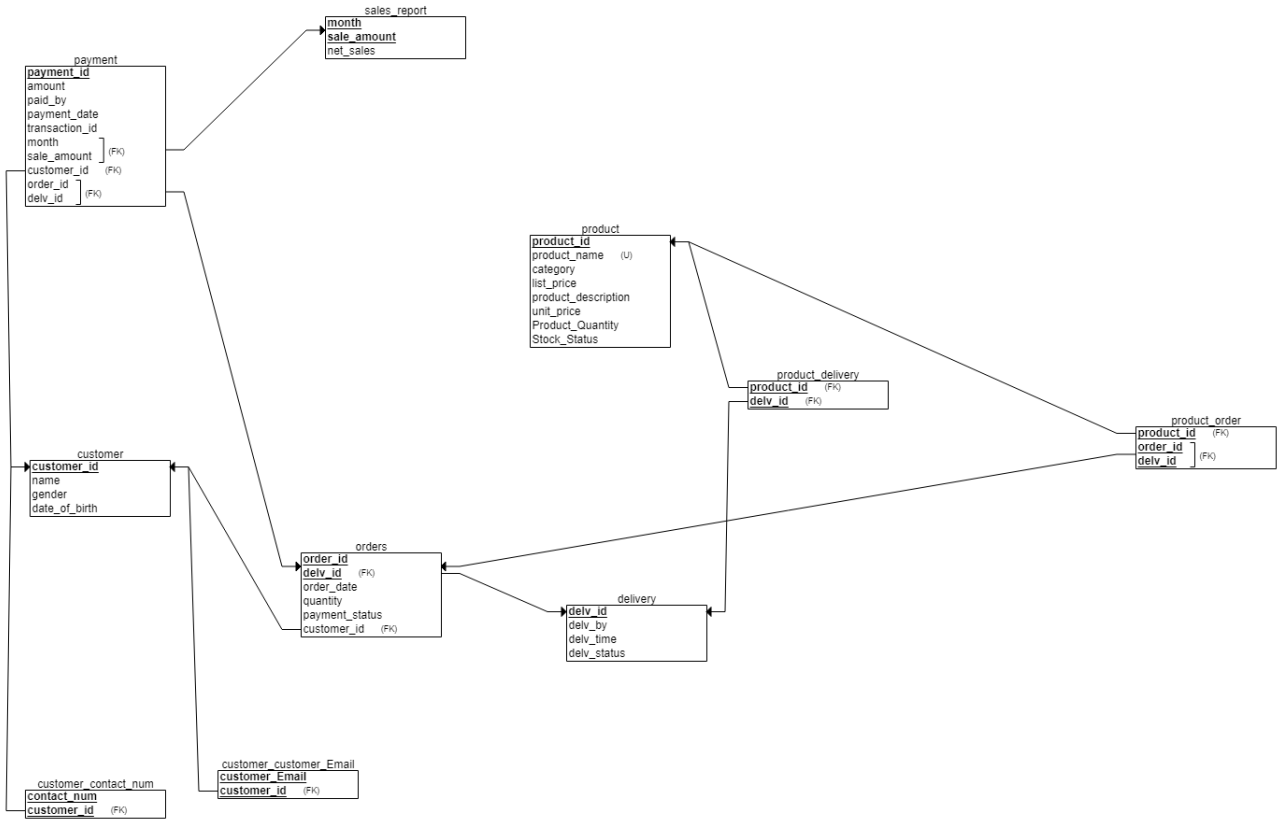
Cardinality constraint: The "Product" entity set to the "Delivery" entity set is one–to–many. This means a product may have many deliveries and delivery may have only one product.

Participation: Product is mandatory for delivery but delivery is optional for the product.

ER Diagram:


name
customer_Email
customer_id
(address)
gender
contact_num
customer
age
date_of_birth
new_order
payment_status
payable
order_id
order_date
quantity
orders
category
product_name
list_price
product_id
product_descriptio
n
product
product_order
unit_price
Stock_Status
Product_Quantity
pay_info
payment_status
product_delivery
order_delivery
amount
payment
paid_by
payment_id
transaction_id
payment_date
sales
delivery
delv_by
delv_id
delv_status
delv_time
sales_report
sale_amount
net_sales
(monthly_sales)
month

Relational Schema:



SQL Schema:

We have attached the SQL schema in a separate file.

Insertion of sample values:

Customer Info Insertion

INSERT INTO `customer`(`customer_id`, `name`, `gender`, `date_of_birth`,address)
VALUES (1,"Fahim Rayhan", "M", 1999-06-12, 1998-11-01,"114 Calhoun St
Smithville, Tennessee(TN), 37166),
(2,"Md.Ahnaf Morshed", "M", 1999-08-18, "74 Mamalahoa Hwy #A
Holualoa, Hawaii(HI), 96725"),

(3,"Md.Adnan Morshed", "M", 1998-08-02, “7364 Us 9 Rte
Elizabethtown, New York(NY), 1293”),
(4,"Fattahil Alom", "M", 1999-07-12,”2841 Lincoln St
Monroe, Louisiana(LA), 71202”),
(5,"Farhin Meem", "F", 1998-02-12,”114 Calhoun St
Smithville, Tennessee(TN), 37166”),
(6,"Bibi Fatema", "F", 1998-11-01,”114 Calhoun St
Smithville, Tennessee(TN), 37166”);

Customer Contact Information

INSERT INTO `customer_contact_num`(`contact_num`, `customer_id`) VALUES
("0160000000",1),
("0190000000",2),
("0190000001",3),
("0150000001",4),
("0170000000",5),
("0180000000",6);

Customer Email Address

INSERT INTO `customer_customer_email`(`customer_Email`, `customer_id`) VALUES
("fahimrayhan@mail.com",1),
("ahnaf@mail.com",2),
("adnan@mail.com",3),
("fattahil@mail.com",4),
("farhin@mail.com",5),
("fatima@mail.com",6),

Product Insertion

INSERT INTO `product` (`product_id`, `product_name`, `category`, `list_price`,
`product_description`, `unit_price`, `Product_Quantity`, `Stock_Status`)
VALUES
('1', 'Galaxy A22', 'Smartphone', '21999', 'Good Smartphone', '17050', '1000', 'Yes'),
('2', 'I Phone 12pro', 'Ios phone', '113000', 'Good Smartphone', '44000', '100', 'Yes'),
('3', 'Galaxy S9', 'Smartphone', '59999', 'Good Smartphone', '22000', '500', 'Yes'),
('4', 'Redmi Note 10', 'Smartphone', '29999', 'Good Smartphone', '15000', '700', 'Yes'),

('5', 'Nova 7i', 'Smartphone', '21999', 'Good Smartphone', '17050', '1000', 'Yes'),
('6', 'Nova 5T', 'Smartphone', '49999', 'Good Smartphone', '26050', '1000', 'Yes');

Delivery Man Creation

```
INSERT INTO `delivery` (`delv_id`, `delv_by`, `delv_time`, `delv_status`) VALUES
('1001', 'Anabil', '09:34:02', '0'),
 ('1002', 'Adil', '08:34:02', '0'),
('1003', 'Shanto', '09:30:02', '0'),
('1004', 'Fardin', '07:34:02', '0'),
('1005', 'Shadhin', '06:34:02', '0'),
('1006', 'Aditto', '09:40:02', '0'),
('1007', 'Kanchon', '09:50:02', '0');
```

Assigning Product to Delivery

```
INSERT INTO `product_delivery` (`product_id`, `delv_id`) VALUES
('1', '1001'),
('2', '1006'),
('3', '1002'),
('4', '1004'),
('6', '1005'),
('5', '1007');
```

Order Creation

```
INSERT INTO `orders` (`order_id`, `quantity`, `payment_status`, `order_date`, `delv_id`,
`customer_id`) VALUES

('11', '5', '0', '2022-01-05', '1002', '1'),
('12', '2', '0', '2022-01-02', '1006', '5'),
('13', '1', '0', '2022-01-03', '1004', '4'),
('14', '2', '0', '2021-12-30', '1005', '3'),
('15', '1', '0', '2021-12-14', '1001', '5'),
('16', '1', '0', '2021-12-15', '1007', '2');
```

Product Order Table

```
INSERT INTO `product_order` (`product_id`, `order_id`, `delv_id`) VALUES
('1', '15', '1001'),
(2, 12, 1006),
(3, 11, 1002),
(4, 13, 1004),
(6, 14, 1005),
(5, 16, 1007);
```

Payment Table

```
INSERT INTO `payment` (`payment_id`, `amount`, `paid_by`, `payment_date`, `transaction_id`,
`month`, `sale_amount`, `customer_id`, `order_id`, `delv_id`)
VALUES
('1', '2000', 'Fahim Rayhan', '2022-01-11', '787787', '2022-01-31', '25000', '1', '11', '1002'),
('2', '5000', 'Ahnaf Morshed', '2022-02-23', '787787', '2022-01-31', '25000', '2', '12', '1001'),
('3', '23000', 'Adnan Morshed', '2022-01-12', '787787', '2022-01-31', '25000', '3', '13', '1003'),
('4', '2800', 'Alif Hassan', '2022-01-12', '787787', '2022-01-31', '25000', '4', '14', '1005'),
('5', '76700', 'Anabil', '2022-01-01', '787787', '2022-01-31', '25000', '5', '15', '1002'),
('6', '45400', 'Jahangir Alam', '2022-01-11', '787787', '2022-01-31', '25000', '6', '16', '1004');
```

GENERATING SALE AMOUNT

```
INSERT INTO `sales_report` (`net_sales`, `month`, `sale_amount`)
VALUES ('127000', '2022-02-28', '2000')
```

Join Queries:

Query1: Finding the customers who have payment due

SQL Statements:

```
SELECT customer_id, name

FROM `customer` NATURAL JOIN orders

WHERE orders.payment_status = 0;
```

Query2: Products that are ready for delivery

SQL Statements:

```
SELECT customer_id, order_id, paid_by, transaction_id,amount, payment_date, delv_id,
delv_by, delv_time
FROM payment NATURAL JOIN delivery
```

Query3: Getting customer full details

SQL Statements:

```
SELECT *
FROM customer NATURAL LEFT OUTER JOIN customer_contact_num
ORDER BY customer.customer_id ASC
```

Query4: Finding the number of orders by a customer

SQL Statements:

```
SELECT customer_id, name, COUNT(order_id) as TotalOrders
FROM customer NATURAL JOIN orders
GROUP BY customer_id;
```

Query5: Finding the delivery man for the specific product

SQL Statements:

```
SELECT product.product_name, delivery.delv_id, delivery.delv_by
FROM delivery NATURAL JOIN product NATURAL JOIN product_delivery
WHERE product.product_name IN (SELECT product_name
FROM product
WHERE product_name = "Galaxy A22")
```

Query6: **Finding Total Product Orders By Category:**

SQL Statements:

```
SELECT  product.category, SUM(orders.quantity) AS Total_Order

FROM product LEFT OUTER JOIN product_order NATURAL JOIN orders ON product.product_id = product_order.product_id

GROUP BY product.category
```

Update & Delete Queries:

Make **some (multiple)** updates and delete queries that must contain **subquery** using IN, NOT IN, EXIST, NOT EXISTS

Query1: Updating Stock

SQL Statement:

```
UPDATE `product`

SET `Stock_Status` = 0

WHERE product_name IN (

        SELECT product_name
```

```
FROM product
WHERE product.Product_Quantity < 4)
```

Query2: Checking Stock Status

SQL Statement:

```
SELECT product_name, Product_Quantity,
CASE WHEN product.Product_Quantity > 4 THEN "In Stock"
ELSE 'The product is out of stock'
END AS QuntityInfo
FROM product
```

Query3: Deleting Customer Who Don’t Have A Phone Number

SQL Statement:

```
DELETE
FROM `customer`
WHERE customer_id NOT IN(
                        SELECT customer_id
                        FROM customer_contact_num)
```

Query4: Alter Table

```
ALTER TABLE `customer_contact_num`
DROP FOREIGN KEY
`customer_contact_num_ibfk_1`;
ALTER TABLE `customer_contact_num`
```

```
ADD CONSTRAINT `customer_contact_num_ibfk_1`
FOREIGN KEY (`customer_id`) REFERENCES `customer`(`customer_id`)
ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `customer_customer_email`
DROP FOREIGN KEY `customer_customer_email_ibfk_1`;
ALTER TABLE `customer_customer_email`
ADD CONSTRAINT `customer_customer_email_ibfk_1`
FOREIGN KEY (`customer_id`) REFERENCES `customer`(`customer_id`)
ON DELETE CASCADE ON UPDATE CASCADE;
```

View and Materialized View:

The view is made for the delivery men. A delivery man should not have the access to the database like delete or update something. There are also some attributes that a delivery man does not need to see like payment status, the birthdate of the customers, and so on. A delivery man should not see the payments, sales report for security concerns.

SQL Query For View

```
CREATE VIEW product_delivery AS

SELECT delivery.delv_id, delivery.delv_by, delivery.delv_time, orders.order_id, orders.quantity, orders.order_date, orders.customer_id, customer.name, customer.address, customer_contact_num.contact_num

FROM delivery NATURAL JOIN product_delivery NATURAL JOIN orders NATURAL JOIN customer NATURAL JOIN customer_contact_num
```

Conclusion:

We have learned how to create a model of a database management system. We have also covered how to design a database management system efficiently. We have got the knowledge to design er-diagram and relational schema of a database. Last but not the least, we have learned about complex SQL joins and subqueries. We can use the model in the e-commerce system and a web application as well.