DATABASE PRESENTATION

DEPARTMENT: SOFTWARE ENGINEERING
PRESENTING TO: MISS AQSA
TOPIC: ONLINE RETAIL STORE

MADE BY:

AASIM NAZIM

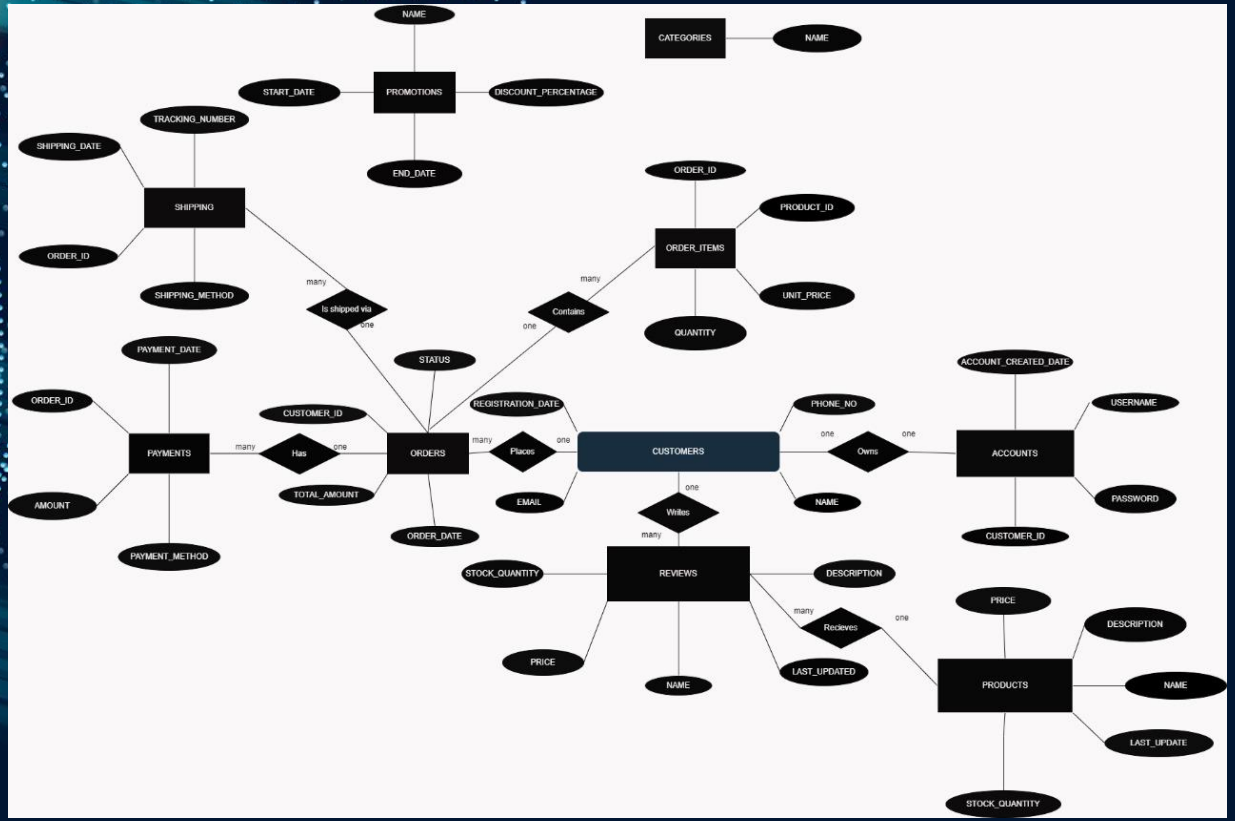
BILAWAL AZEEM

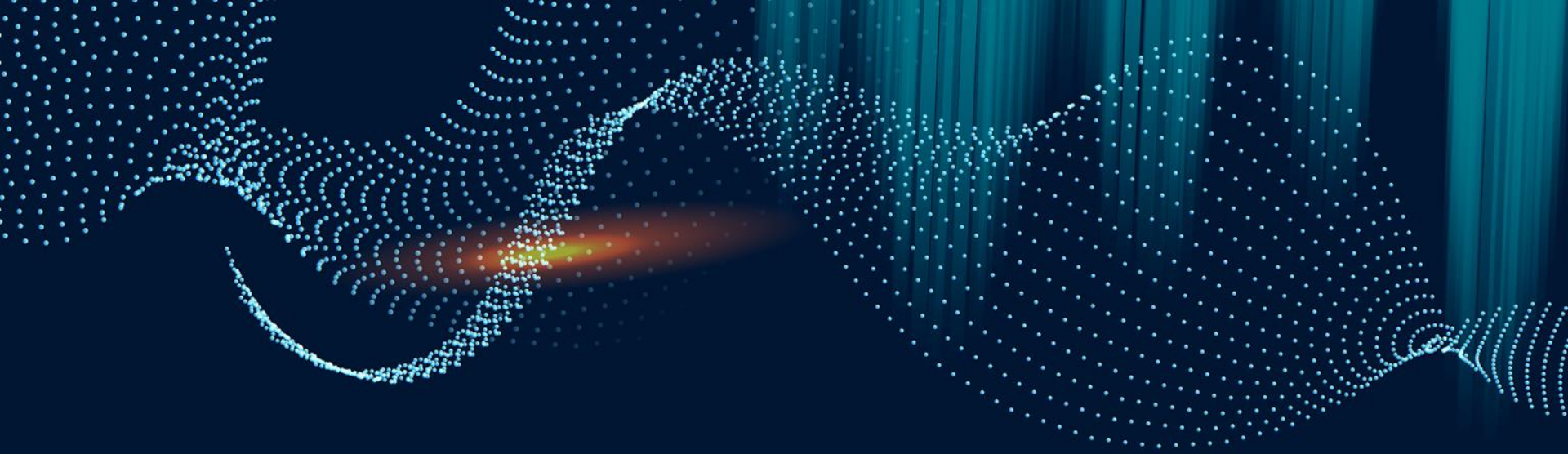
HUZAIFA DILDAR

ONLINE RETAIL STORE

An online retail store is a digital platform or website where consumers can browse, select, and purchase products or services over the internet. These stores provide a virtual shopping experience, often offering a wide range of items including electronics, clothing, books, and more.

ER MODEL


NAME
CATEGORIES
NAME
START_DATE
PROMOTIONS
DISCOUNT_PERCENTAGE
TRACKING_NUMBER
SHIPPING_DATE
END_DATE
SHIPPING
ORDER_ID
PRODUCT_ID
ORDER_ITEMS
ORDER_ID
SHIPPING_METHOD
many
Is shipped via
one
many
Contains
one
UNIT_PRICE
QUANTITY
PAYMENT_DATE
STATUS
ACCOUNT_CREATED_DATE
ORDER_ID
CUSTOMER_ID
REGISTRATION_DATE
PHONE_NO
USERNAME
PAYMENTS
many
Has
one
ORDERS
many
Places
one
CUSTOMERS
one
Owns
one
ACCOUNTS
AMOUNT
TOTAL_AMOUNT
EMAIL
one
Writes
NAME
PASSWORD
ORDER_DATE
CUSTOMER_ID
PAYMENT_METHOD
many
STOCK_QUANTITY
REVIEWS
DESCRIPTION
PRICE
many
Recieves
one
DESCRIPTION
PRICE
NAME
LAST_UPDATED
PRODUCTS
NAME
LAST_UPDATE
STOCK_QUANTITY



CONTRIBUTION

ADDED BY HUZAIFA

```sql
#Queries#
-- Query 1: Retrieve customer details along with their orders
SELECT c.name, c.email, o.order_id, o.order_date
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id;

-- Query 2: Retrieve product details along with their categories
SELECT p.name AS product_name, p.price, c.name AS category_name
FROM Products p
JOIN Categories c ON p.category_id = c.category_id;

-- Query 3: Count the number of orders for each customer
SELECT c.name, COUNT(o.order_id) AS num_orders
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id;

-- Query 4: Retrieve orders with total amount greater than average total amount
SELECT *
FROM Orders
WHERE total_amount > (SELECT AVG(total_amount) FROM Orders);

-- Query 5: List unique categories
SELECT DISTINCT name
FROM Categories;
```

```sql
-- Query 6: Retrieve order details along with product information
SELECT o.order_id, oi.product_id, p.name AS product_name, oi.quantity, oi.unit_price
FROM Orders o
JOIN Order_Items oi ON o.order_id = oi.order_id
JOIN Products p ON oi.product_id = p.product_id;

-- Query 7: Retrieve orders placed in May 2024
SELECT *
FROM Orders
WHERE MONTH(order_date) = 5 AND YEAR(order_date) = 2024;

-- Query 8: Search for customers with 'John' in their name
SELECT *
FROM Customers
WHERE name LIKE '%John%';

-- Query 9: Find the latest product update date
SELECT MAX(last_updated) AS latest_update
FROM Products;

-- Query 10: Retrieve top 5 best-selling products
SELECT product_id, SUM(quantity) AS total_sold
FROM Order_Items
GROUP BY product_id
ORDER BY total_sold DESC
```

```sql
-- Query 11: Retrieve detailed order information including customer and product details
SELECT o.order_id, c.name AS customer_name, p.name AS product_name, oi.quantity, oi.unit_price
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
JOIN Order_Items oi ON o.order_id = oi.order_id
JOIN Products p ON oi.product_id = p.product_id;

-- Query 12: Retrieve orders with formatted order dates
SELECT order_id, DATE_FORMAT(order_date, '%Y-%m-%d') AS formatted_order_date
FROM Orders;

-- Query 13: Count the number of unique customers
SELECT COUNT(DISTINCT customer_id) AS num_customers
FROM Orders;

-- Query 14: Find customers who have not placed any orders
SELECT *
FROM Customers
WHERE customer_id NOT IN (SELECT DISTINCT customer_id FROM Orders);

-- Query 15: Find customers with more than 2 orders
SELECT customer_id, COUNT(*) AS num_orders
FROM Orders
GROUP BY customer_id
HAVING num_orders > 2;
```

```sql

-- Query 16: Retrieve payment details along with order information
SELECT o.order_id, o.order_date, p.payment_date, p.payment_method, p.amount
FROM Orders o
JOIN Payments p ON o.order_id = p.order_id;


-- Query 18: Retrieve orders along with payment information (including those without payments)
SELECT o.order_id, o.order_date, p.payment_date, p.payment_method, p.amount
FROM Orders o
LEFT JOIN Payments p ON o.order_id = p.order_id;
```

ADDED BY BILAWAL

```sql
-- Query 19: Calculate total sales amount for each product
SELECT oi.product_id, p.name AS product_name, SUM(oi.quantity * oi.unit_price) AS total_sales
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
GROUP BY oi.product_id;

-- Query 20: Calculate delivery date by adding 3 days to order date
SELECT order_id, order_date, DATE_ADD(order_date, INTERVAL 3 DAY) AS delivery_date
FROM Orders;

-- Query 21: Categorize orders as 'High' if total amount > 50, 'Medium' if > 20, else 'Low'
SELECT order_id, total_amount,
    CASE
        WHEN total_amount > 50 THEN 'High'
        WHEN total_amount > 20 THEN 'Medium'
        ELSE 'Low'
    END AS order_category
FROM Orders;

-- Query 22: Find customers who have submitted reviews
SELECT DISTINCT c.customer_id, c.name
FROM Customers c
WHERE EXISTS (
    SELECT 1
    FROM Reviews r
    WHERE r.customer_id = c.customer_id
);

-- Query 23: Round product prices to two decimal places
SELECT name, ROUND(price, 2) AS rounded_price
FROM Products;

-- Query 24: Calculate average product rating
SELECT product_id, AVG(rating) AS avg_rating
FROM Reviews
GROUP BY product_id;

-- Query 25: Retrieve orders placed by customers named 'Alice'

-- Query 26: Count the number of unique products sold
SELECT COUNT(DISTINCT product_id) AS num_products_sold
FROM Order_Items;

-- Query 27: Find the oldest and newest order dates
SELECT MIN(order_date) AS oldest_order_date, MAX(order_date) AS newest_order_date
FROM Orders;


-- Query 29: Extract first 3 characters of product name
SELECT product_id, SUBSTRING(name, 1, 3) AS short_name
FROM Products;

FROM Products;

-- Query 30: Concatenate product names for each order
SELECT order_id, GROUP_CONCAT(p.name SEPARATOR ', ') AS product_names
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
GROUP BY order_id;

-- Query 31: Calculate total sales for May 2024
SELECT SUM(oi.quantity * oi.unit_price) AS total_sales
FROM Order_Items oi
JOIN Orders o ON oi.order_id = o.order_id
WHERE YEAR(o.order_date) = 2024 AND MONTH(o.order_date) = 5;

-- Query 32: Combine results of two queries
(SELECT product_id, name FROM Products WHERE price > 50)
UNION
(SELECT product_id, name FROM Products WHERE stock_quantity < 10);

-- Query 33: Retrieve the latest review for each product
SELECT p.product_id, p.name AS product_name, r.rating, r.review_text, r.review_date
FROM Products p
JOIN Reviews r ON p.product_id = r.product_id
ORDER BY r.review_date DESC;

-- Query 34: Retrieve orders placed by specific customers
SELECT *
FROM Orders
WHERE customer_id IN (1, 2, 3);
```

ADDED BY AASIM

```sql
-- Query 35: Retrieve top 3 customers with highest total order amount
SELECT c.customer_id, c.name, SUM(o.total_amount) AS total_order_amount
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
ORDER BY total_order_amount DESC
LIMIT 3;
-- Query 36: Retrieve the top 3 most recent orders
SELECT *
FROM Orders
ORDER BY order_date DESC
LIMIT 3;

-- Query 37: Retrieve products with a price greater than $50 and a stock quantity less than 10
SELECT *
FROM Products
WHERE price > 1000 AND stock_quantity < 20;

-- Query 38: Retrieve orders placed on weekends (Saturday or Sunday)
SELECT *
FROM Orders
WHERE DAYOFWEEK(order_date) IN (1, 7);
-- Query 39: Retrieve products with a description longer than 100 characters
SELECT *
FROM Products
```

```sql
WHERE LENGTH(description) > 20;

-- Query 40: Convert product names to uppercase
SELECT UPPER(name) AS upper_product_name
FROM Products;

-- Query 41: Convert customer names to lowercase
SELECT LOWER(name) AS lower_customer_name
FROM Customers;

-- Query 42: Concatenate customer names and email addresses
SELECT CONCAT(name, ' - ', email) AS customer_info
FROM Customers;

-- Query 43: Retrieve the index of the '@' symbol in email addresses
SELECT email, LOCATE('@', email) AS at_symbol_index
FROM Customers;

-- Query 44: Retrieve the index of the first occurrence of 'o' in product names
SELECT name, LOCATE('o', name) AS o_index
FROM Products;

-- Query 45: Retrieve products with an average rating greater than 4
SELECT product_id, AVG(rating) AS avg_rating
FROM Reviews
GROUP BY product_id
```

```sql
FROM Reviews
GROUP BY product_id
HAVING avg_rating > 4;

-- Query 46: Retrieve products that have been ordered more than 5 times
SELECT product_id, COUNT(order_item_id) AS times_ordered
FROM Order_Items
GROUP BY product_id
HAVING times_ordered > 5;

-- Query 47: Retrieve the count of reviews per product
SELECT product_id, COUNT(review_id) AS num_reviews
FROM Reviews
GROUP BY product_id;

-- Query 48: Retrieve the average rating per product
SELECT product_id, AVG(rating) AS avg_rating
FROM Reviews
GROUP BY product_id;

-- Query 49: Retrieve the count of reviews per product
SELECT product_id, COUNT(review_id) AS num_reviews
FROM Reviews
GROUP BY product_id;
```

```sql
-- Query 50: Retrieve payments made using 'Credit Card'
SELECT payment_id FROM Payments WHERE payment_method = 'Credit Card';

-- Query 51: Retrieve reviews with a rating of 5
SELECT review_id FROM Reviews WHERE rating = 5;
```

EXPLANATION AND OUTPUT

SOME QUERIES AND THEIR OUTPUT

INPUT

```sql
-- Query 6: Retrieve order details along with product information
SELECT o.order_id, oi.product_id, p.name AS product_name, oi.quantity, oi.unit_price
FROM Orders o
JOIN Order_Items oi ON o.order_id = oi.order_id
JOIN Products p ON oi.product_id = p.product_id;
```

OUTPUT

Result Grid | Filter Rows: | Export: | Wra

order_id	product_id	product_name	quantity	unit_price
1	1	Product A	2	19.99
2	2	Product B	2	29.99
3	3	Product C	3	9.99
4	4	Product D	3	14.99

```sql
-- Query 11: Retrieve detailed order information including customer and product details
SELECT o.order_id, c.name AS customer_name, p.name AS product_name, oi.quantity, oi.unit_price
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
JOIN Order_Items oi ON o.order_id = oi.order_id
JOIN Products p ON oi.product_id = p.product_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_name	product_name	quantity	unit_price
1	Alice Smith	Product A	2	19.99
2	Bob Johnson	Product B	2	29.99
3	Eva Brown	Product C	3	9.99
4	David Lee	Product D	3	14.99

```sql
-- Query 10: Retrieve top 5 best-selling products
SELECT product_id, SUM(quantity) AS total_sold
FROM Order_Items
GROUP BY product_id
ORDER BY total_sold DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap C

product_id	total_sold
3	3
4	3
1	2
2	2

Refresh data re-executing the original query

SOME QUERIES AND THEIR OUTPUT

INPUT

```sql
-- Query 21: Categorize orders as 'High' if total amount > 50, 'Medium' if > 20, else 'Low'
SELECT order_id, total_amount,
    CASE
        WHEN total_amount > 50 THEN 'High'
        WHEN total_amount > 20 THEN 'Medium'
        ELSE 'Low'
    END AS order_category
FROM Orders;
```

OUTPUT

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	total_amount	order_category
1	39.98	Medium
2	59.98	High
3	29.97	Medium
4	44.97	Medium

```sql
-- Query 30: Concatenate product names for each order
SELECT order_id, GROUP_CONCAT(p.name SEPARATOR ', ') AS product_names
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
GROUP BY order_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	product_names
1	Product A
2	Product B
3	Product C
4	Product D

```sql
-- Query 33: Retrieve the latest review for each product
SELECT p.product_id, p.name AS product_name, r.rating, r.review_text, r.review_date
FROM Products p
JOIN Reviews r ON p.product_id = r.product_id
ORDER BY r.review_date DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	rating	review_text	review_date
4	Product D	5	Awesome product, highly recommended	2024-05-25 12:00:00
1	Product A	5	Excellent product!	2024-05-25 10:00:00
2	Product B	4	Great product, fast delivery	2024-05-24 14:00:00
3	Product C	3	Average product, could be better	2024-05-23 11:00:00

SOME QUERIES AND THEIR OUTPUT

INPUT

```sql
-- Query 35: Retrieve top 3 customers with highest total order amount
SELECT c.customer_id, c.name, SUM(o.total_amount) AS total_order_amount
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
ORDER BY total_order_amount DESC
LIMIT 3;
```

OUTPUT

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

customer_id	name	total_order_amount
2	Bob Johnson	59.98
4	David Lee	44.97
1	Alice Smith	39.98

```sql
-- Query 45: Retrieve products with an average rating greater than 4
SELECT product_id, AVG(rating) AS avg_rating
FROM Reviews
GROUP BY product_id
HAVING avg_rating > 4;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	avg_rating
1	5.0000
4	5.0000

```sql
-- Query 49: Retrieve the count of reviews per product
SELECT product_id, COUNT(review_id) AS num_reviews
FROM Reviews
GROUP BY product_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	num_reviews
1	1
2	1
3	1
4	1

THANKS!