

MYSQL Project - Ecommerce Platform

Project Overview

In today's digital era, e-commerce has become an integral part of the global economy, enabling consumers to shop for a wide range of products and services conveniently from their mobile devices. This report presents a comprehensive overview of our e-commerce mobile application project, a venture aimed at delivering a seamless and user-friendly shopping experience to our customers.

To Develop the E-commerce application we have used the following SQL tables mentioned below:

- User Authentication
- Products
- Categories
- Seller
- Payment
- Deliveries
- Transaction Receipt
- Shipping Order

Project Analysis and Requirements

User Management: Register, login, update profile, password reset.

Product Management: Add, remove, update product information.

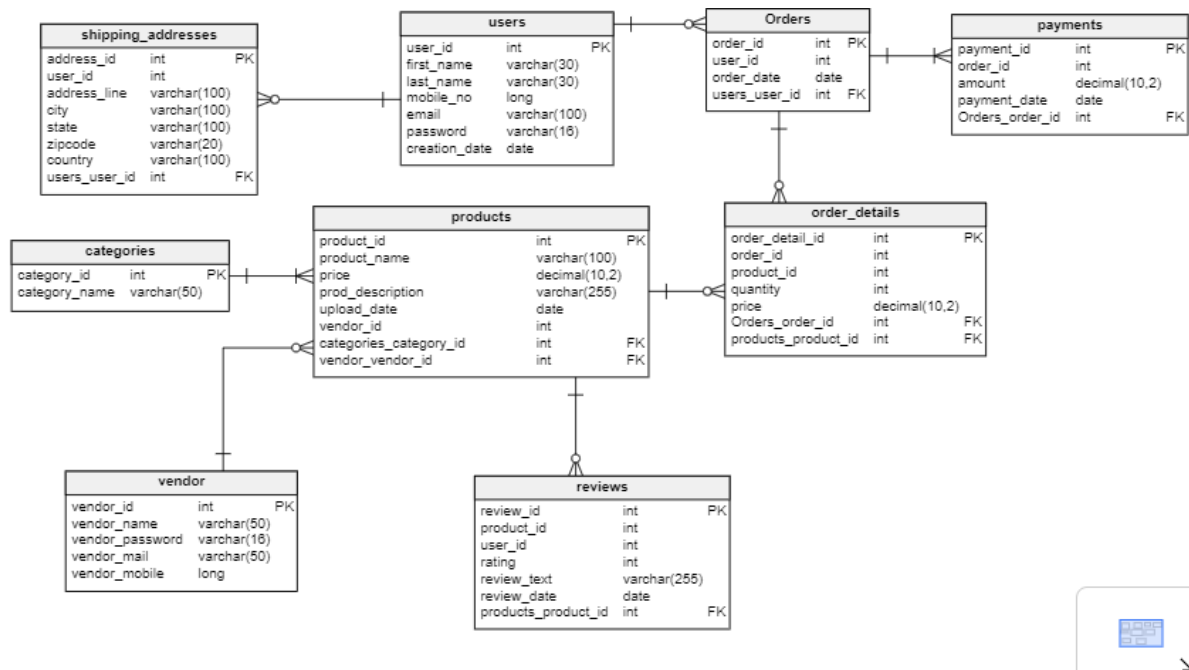
Order Processing: Allow users to add items to their cart, place orders, and view order history.

Reviews: Let customers leave reviews and ratings on products.

Category Navigation: Allow users to filter products based on categories.

Payment Handling: Process payments securely.

ER Diagram



Normalization

Normalization in MySQL involves structuring a database to minimize redundancy and dependency issues while maintaining data integrity. Here's a general guideline on how you might normalize a database in MySQL:

Steps for Normalization:

First Normal Form (1NF):

Ensure atomic values in each field.

Split columns with multiple values into separate fields.

Avoid repeating groups within a single row.

Second Normal Form (2NF):

Ensure each non-key column is fully dependent on the entire primary key.

If a table has a composite primary key, consider creating new tables to handle dependent attributes.

Third Normal Form (3NF):

Eliminate transitive dependencies.

If non-prime attributes depend on other non-prime attributes, create separate tables to resolve these dependencies.

Further Normalization (BCNF, 4NF, 5NF):

BCNF (Boyce-Codd Normal Form), 4NF, and 5NF are further stages of normalization that resolve additional specific types of anomalies. However, not all databases require these higher forms of normalization.

Table Creation:

User Authentication Table

- User_id
- First_name
- Last_name
- Mobile
- email
- Password
- Creation_Date

```
create database OneStore;
use OneStore;

create table users(
  user_id int primary key,
  first_name varchar(30),
  last_name varchar(30),
  mobile_no long,
  email varchar(100),
  password varchar(16),
  creation_date date);

insert into users values(1,'Ravi','Krishna',7386847872,'ravi2000@gmail.com','Ravi@2000','2000-07-19');
insert into users values(2, 'Sara', 'Johnson', 9876543210, 'sara.johnson@gmail.com', 'Sara@123', '2000-08-15');
insert into users values(3, 'John', 'Doe', 8765432109, 'john.doe@gmail.com', 'Doe@456', '2000-09-05');
insert into users values(4, 'Emily', 'Smith', 7654321098, 'emily.smith@gmail.com', 'Smith@789', '2000-10-12');
insert into users values(5, 'Michael', 'Johnson', 6543210987, 'michael.j@gmail.com', 'Michael@987', '2000-10-12');
insert into users values(6, 'Reddy', 'Rao', 8499923059, 'reddyr Rao@gmail.com', 'Reddy@200', '2000-05-12');
```

Vendor

- Vendor_id
- Vendor_name
- Vendor_password
- Vendor_mail
- Vendor_phone_number

```
create table vendor(  
  vendor_id int primary key,  
  vendor_name varchar(50),  
  vendor_password varchar(16),  
  vendor_mail varchar(50),  
  vendor_mobile long);  
  
insert into vendor values(1,'Venkataiah','venky@1980','venky1980@gmail.com',8889992387);  
insert into vendor values(2,'Srinivas','srinivas@123','srinivas@example.com',7778883456);  
insert into vendor values(3,'Ananya','ananya@456','ananya@example.com',9998882678);  
insert into vendor values(4,'Prakash','prakash@789','prakash@example.com',6667771234);  
insert into vendor values(5,'Neha','neha@abc','neha@example.com',8887775432);  
insert into vendor values(6,'Amit','amit@xyz','amit@example.com',5556668765);
```

Orders

- Order_id
- User_id
- Product_id
- Order_date

```

CREATE TABLE orders (
    order_id int PRIMARY KEY,
    user_id INT,
    product_id int,
    order_date date,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);

insert into orders values(1,1,'2023-08-11');
insert into orders values(2, 2, '2023-08-12');
insert into orders values(3, 3, '2023-08-13');
insert into orders values(4, 4, '2023-08-14');
insert into orders values(5, 5, '2023-08-15');
insert into orders values(6, 6,'2023-12-05');

```

Order_details

- Order_detail_id
- Order_id
- Product_id
- User_id
- Quantity
- Price

```

CREATE TABLE order_details (
    order_detail_id INT PRIMARY KEY,
    order_id INT,
    product_id INT,
    user_id int,
    quantity INT,
    price DECIMAL(10, 2),
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);

insert into order_details values (1,1,1,1,40000);
insert into order_details values (2, 2, 2, 2, 15000);
insert into order_details values (3, 3, 3, 3, 18000);
insert into order_details values (4, 4, 4, 4, 11000);
insert into order_details values (5, 5, 5, 5, 13500);
insert into order_details values (6,6,6,6,16000);

```

Products Table

- Product_id
- Product_name
- Prod_description
- Price
- Upload_date
- Vendor_id

```
CREATE TABLE products (  
  product_id int PRIMARY KEY,  
  product_name VARCHAR(100),  
  price DECIMAL(10, 2),  
  prod_description varchar(240),  
  upload_date date,  
  vendor_id int,  
  FOREIGN KEY (vendor_id) REFERENCES vendor(vendor_id)  
);  
  
insert into products values (1, 'Samsung', 9999.99, 'Mobile with dual camera', '2023-08-11', 1);  
insert into products values(2, 'Apple iPhone', 1099.99, 'Latest iPhone model with advanced features', '2023-08-12', 2);  
insert into products values(3, 'HP Pavilion Laptop', 799.99, 'Powerful laptop for gaming and work', '2023-08-13', 3);  
insert into products values(4, 'Sony 55-inch Smart TV', 899.99, 'High-quality smart TV with 4K resolution', '2023-08-14', 4);  
insert into products values(5, 'Adidas Running Shoes', 79.99, 'Comfortable shoes for running and sports', '2023-08-15', 5);  
insert into products values(6, 'Samsung Refrigerator', 699.99, 'Energy-efficient refrigerator with large storage', '2023-08-16', 6);  
|
```

Categories

- Category_id
- Category_name

```
CREATE TABLE categories (  
  category_id INT AUTO_INCREMENT PRIMARY KEY,  
  category_name VARCHAR(50)  
);  
  
insert into categories values (1, 'Mobiles');  
insert into categories values(2, 'Laptops');  
insert into categories values(3, 'Electronics');  
insert into categories values(4, 'Clothing');  
insert into categories values(5, 'Home Appliances');  
insert into categories values(6, 'Books');  
|
```

Reviews

- Review_id
- Product_id
- User_id
- Rating
- Review_Text
- Review_Date

```
CREATE TABLE reviews (  
    review_id INT PRIMARY KEY,  
    product_id INT,  
    user_id INT,  
    rating INT,  
    review_text varchar(255),  
    review_date date,  
    FOREIGN KEY (product_id) REFERENCES products(product_id),  
    FOREIGN KEY (user_id) REFERENCES users(user_id)  
);  
  
insert into reviews values(1,1,1,9,'Good Product','2023-08-11');  
insert into reviews values(2, 2, 2, 8, 'Satisfactory purchase', '2023-08-12');  
insert into reviews values(3, 3, 3, 8, 'Nice product, fast delivery', '2023-08-13');  
insert into reviews values(4, 4, 4, 7, 'Average quality, needs improvement', '2023-08-14');  
insert into reviews values(5, 5, 5, 9, 'Excellent service, highly recommended', '2023-08-15');  
insert into reviews values(6, 6, 6, 7, 'Good value for money','2023-08-9');
```

Shipping Address

- Address_id
- User_id
- Address_line
- City
- State
- Zip_code
- Country

```

CREATE TABLE shipping_addresses (
    address_id INT PRIMARY KEY,
    user_id INT,
    address_line VARCHAR(255),
    city VARCHAR(100),
    state VARCHAR(100),
    zip_code VARCHAR(20),
    country VARCHAR(100),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);

insert into shipping_addresses values (1,1,'2-3,Kanaka Street','Mysore','Karnataka',457865,'India');
insert into shipping_addresses values (2, 2, '5-7/A, Tulip Road', 'Bangalore', 'Karnataka', 560001, 'India');
insert into shipping_addresses values (3, 3, '8-10/2, Rose Lane', 'Chennai', 'Tamil Nadu', 600001, 'India');
insert into shipping_addresses values (4, 4, '12/7, Lily Street', 'Hyderabad', 'Telangana', 500001, 'India');
insert into shipping_addresses values (5, 5, '18-20/K, Orchid Avenue', 'Mumbai', 'Maharashtra', 400001, 'India');
insert into shipping_addresses values (6, 6, '22-3/B, Sunflower Lane', 'Delhi', 'Delhi', 110001,'India');

```

Payment

- Payment_id
- Order_id
- Payment_method
- Amount
- Payment_date

```

CREATE TABLE payments (
    payment_id INT PRIMARY KEY,
    order_id INT,
    amount DECIMAL(10, 2),
    payment_date date,
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
);

insert into payments values (1,1,9999.99,'2023-11-08');
insert into payments values (2, 2, 799.99, '2023-11-09');
insert into payments values (3, 3, 1499.50, '2023-11-10');
insert into payments values (4, 4, 399.95, '2023-11-11');
insert into payments values (5, 5, 199.99, '2023-11-12');
insert into payments values (6, 6, 599.75, '2023-11-13');

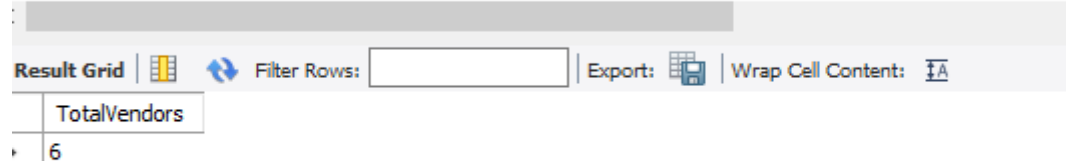
```


Write queries to help answer business and customer question. These are the types of reports you will be asked to build in real world scenarios. So use your creativity to answer the question in the best possible way based on your specific database design. Showcase use of functions and database concepts you learned in the class.

Business Questions-

1. How many vendors do you have registered?

```
154 • select count(vendor_id) as TotalVendors from vendor;
155
```

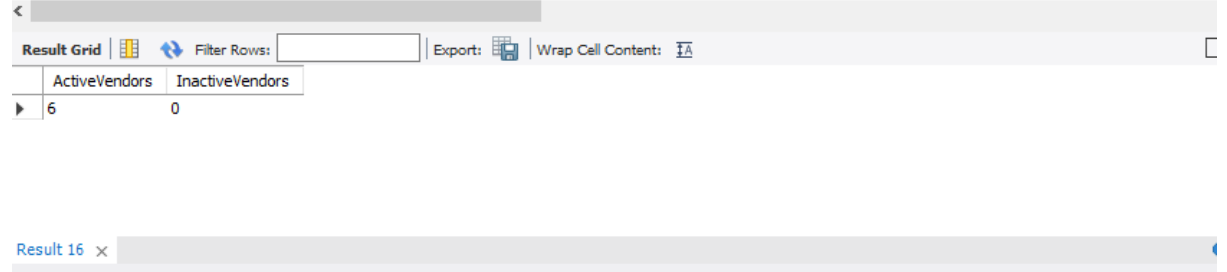


The screenshot shows a SQL query in a text editor with line numbers 154 and 155. The query is `select count(vendor_id) as TotalVendors from vendor;`. Below the editor is a toolbar with 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content' options. The 'Result Grid' is active, displaying a table with one column 'TotalVendors' and one row with the value '6'.

TotalVendors
6

2. Number of active vs inactive vendors are there with the company (this month)?

```
161 • Select
162 sum(case when year(current_date) = year(order_date) and month(current_date) = month(order_date) then 1 else 0 end) as ActiveVendc
163 sum(case when year(current_date) != year(order_date) or month(current_date) != month(order_date) then 1 else 0 end) as InactiveVer
164 from vendor v
165 join order_details o on v.product_id=o.product_id;
```



The screenshot shows a SQL query in a text editor with line numbers 161 through 165. The query is `Select sum(case when year(current_date) = year(order_date) and month(current_date) = month(order_date) then 1 else 0 end) as ActiveVendc sum(case when year(current_date) != year(order_date) or month(current_date) != month(order_date) then 1 else 0 end) as InactiveVer from vendor v join order_details o on v.product_id=o.product_id;`. Below the editor is a toolbar with 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content' options. The 'Result Grid' is active, displaying a table with two columns 'ActiveVendors' and 'InactiveVendors'. The 'ActiveVendors' column has a value of '6' and the 'InactiveVendors' column has a value of '0'.

ActiveVendors	InactiveVendors
6	0

Result 16 ×

3. How many customers have registered with your platform from the beginning?

163

```
164 select count(user_id) as TotalCustomers from users;
```

Result Grid	
Filter Rows:	
Export:	Wrap Cell Content:
	TotalCustomers
▶	6

Default 7

4. Who is the vendor with number listings?

```
2 • select count(*) num,vendor_name
3   from products p
4   join vendor v on v.vendor_id=p.vendor_id
5   group by vendor_name
6   order by num desc
```

Result Grid	
Filter Rows:	
Export:	Wrap Cell Content:
num	vendor_name
1	Venkataiah

5. Who is the customer with most number of orders?

```
6 • select count(*) num,first_name
7   from orders o
8   join users u on u.user_id=o.user_id
9   group by first_name
```

Result Grid	
Filter Rows:	
Export:	Wrap Cell Content:
num	first_name
1	Ravi

6. Top 5 vendors by revenue by month- January, Feb, and Mar?

```
228 • select
229     v.vendor_name,
230     sum(case when month(o.order_date)=1 then o.price else 0 end) as r1,
231     sum(case when month(o.order_date)=2 then o.price else 0 end) as r2,
232     sum(case when month(o.order_date)=3 then o.price else 0 end) as r3
233 from order_details o
234 join vendor v on o.product_id=v.product_id
235 group by vendor_name
236 order by (r1+r2+r3) desc
237 limit 5;
```

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

	vendor_name	r1	r2	r3
▶	Venkataiah	0.00	0.00	0.00
	Srinivas	0.00	0.00	0.00
	Ananya	0.00	0.00	0.00
	Prakash	0.00	0.00	0.00
	Neha	0.00	0.00	0.00

```
191     order by (r1+r2+r3) desc
192     limit 3;
```

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

	vendor_name	r1	r2	r3
▶	Venkataiah	0.00	0.00	0.00
	Srinivas	0.00	0.00	0.00
	Ananya	0.00	0.00	0.00

Result 17 x

7. Top5 customers by revenue this year?

```
239 • select first_name,sum(price) as rev
240 from users u
241 inner join order_details o on u.user_id=o.user_id
242 where year(order_date)=year(current_date())
243 group by first_name
244 order by rev desc
245 limit 5;
```

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

	first_name	rev
▶	Ravi	40000.00
	John	18000.00
	Reddy	16000.00
	Sara	15000.00
	Michael	13500.00

Result 26 x

8. Has our revenue increased last month compared to last year same month?

```
238
239 • select
240     year(curdate()) as current_year, month(curdate()) as current_month,
241     sum(case when year(order_date)=year(curdate()) then price else 0 end) as cur_year_revenue,
242     sum(case when year(order_date)=year(curdate())-1 and month(order_date)= month(curdate()) then price else 0 end) as last_year_revenue
243 from order_details;
```

Result Grid

	current_year	current_month	cur_year_revenue	last_year_revenue
▶	2023	11	113500.00	0.00

Customer questions-

9. what is most expensive order among my orders?

```
243
244 • select order_id, price from order_details order by price desc limit 1;
```

Result Grid

	order_id	price
▶	1	40000.00

10. How many orders did I place with your business?

```
249 • select count(*) from order_details where user_id=1
```

Result Grid

	count(*)
▶	1

11. Total historical spends with the business year to date?

```
251 • select sum(price) as total_price
252 from order_details
253 where year(order_date)=year(current_date());
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	total_price
▶	113500.00

12. Present the customer with a trend chart of their expenditure, You can use excel for this question, if needed. Use data from your database.

```
245 • select u.first_name,
246 o.order_date,
247 o.price
248 from users u join order_details o on u.user_id=o.user_id
249 order by u.first_name,o.order_date;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	first_name	order_date	price
▶	Emily	2023-11-08	11000.00
	John	2023-11-08	18000.00
	Michael	2023-11-08	13500.00
	Ravi	2023-11-08	40000.00
	Reddy	2023-11-08	16000.00
	Sara	2023-11-08	15000.00

Operations questions-

13. what is the size of my database? (research database size for MYSQL)

```

255 • select round(sum(data_length+index_length)/1024/1024,2) as MB from information_schema.tables
256 where table_schema='OneStore';

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	MB			
▶	0.31			

14. In any of above queries can you change your database design an reduce the cost of the query?

- Denormalization:

In some cases, you might consider denormalizing specific tables to reduce the need for complex joins.

For read-heavy operations, denormalization can improve query performance, even if it slightly increases data redundancy.

- Caching:

Implement caching mechanisms to store frequently accessed data in memory. Caching can reduce the need for repeated database queries and improve response times for common requests.

- Materialized Views:

Consider using materialized views to precompute and store the results of complex queries.

Materialized views can speed up data retrieval for frequently used aggregations or reports.