APPLE STORE PROJECT

PROJECT DESCRIPTION:

Apple Store is a database that stores the information about the apple products, registered user information and transaction details. The database will contain information of only apple products which can be divided into 2, 3 sub categories like laptops, iPhone, iPad, accessories. The Database can be ecommerce platforms specifically selling apple products like Apple.com website or any reseller like Best Buy.

The Database will be addressing the following objectives to store the following.

- The details about each product ex: productid, type, memory.
- The information about the users ex: name, address, payment details.
- Order details of a particular product order ex ordered, payment type, delivery information
- Inventory details of the products
- Product Category of the products
- Promotional offers which are running on a product.

Apple stores are considerably changed the landscape for consumer electronics retailers and influenced other technological companies to follow suit. The items themselves are typically of higher quality, the salespeople at Apple are typically more competent, and the storefronts are normally more aesthetically and functionally appealing.

OBJECTIVES:

- To provide good products and great services to enrich lives of people
- To make users highly engaged and satisfied with great customer service
- To provide the students and other customers with the greatest personal computing experience possible.
- To make pre-orders and shipped out.
- To check the promotional offers present on the product

SCOPE:

To Ease Day to Day in-store operations of an apple store, which would expedite the inventory information and help them in managing their business more efficiently.

PROJECT REQUIREMENTS:

- ✓ **Operating System**: Windows
- ✓ **Database**: PhpMyadmin
- ✓ **Applications**: Microsoft word

DATABASE REQUIREMENTS:

The following information contains the data tables for the collection of data base**.**

1. User's information table
2. Orders table
3. Payments of the products table
4. Products table
5. Product category table
6. Order items table
7. Product inventory table

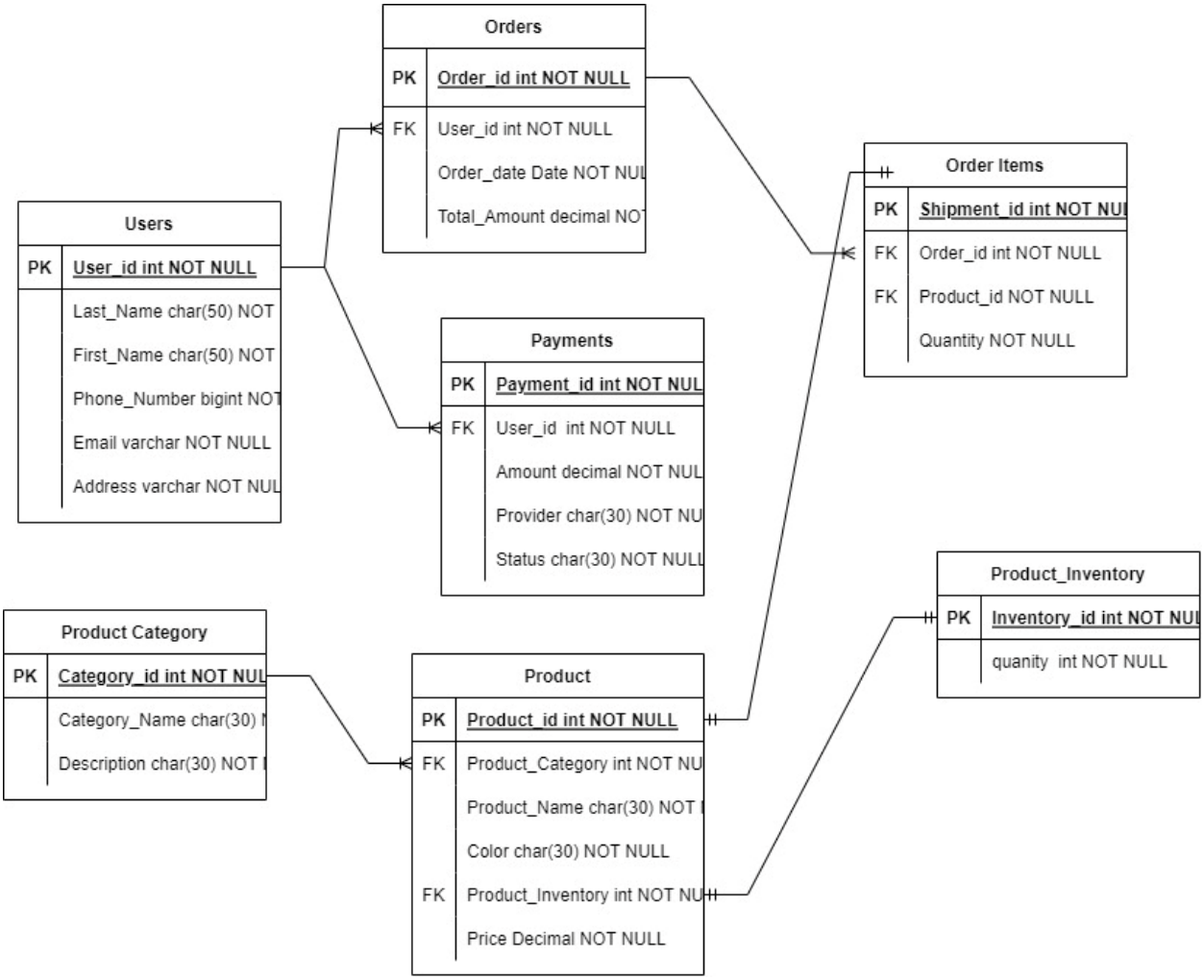
USER REQUIREMENTS:

- The database will let the user to add, delete, select, update the product details and make changes to the inventory.
- Only specific users have the rights to do the above transactions.
- Store should have multiple profiles for access like manager, employee, guest.
- The manager login should act like a super user with read, write and view access to all the data whereas an employee login should be restricted only to the read, write and view access to selected data and a guest login should have only selected view access without any read or write.
- The Database should be able to handle multiple users at a time.

BUSINESS RULES:

1) As per our assumptions, we have designed the database
2) To make a purchase a user should be registered with store.
3) After a product is purchased, it should be removed from the inventory.
4) Each product should have a unique Product_Id.
5) It should only contain apple products.
6) A user can buy only max 3 products of a particular category.
7) Each user should have unique phone number.
8) The quantity in orders should be less than or equal to quantity in inventory.
9) Order should be delivered only after the payment is approved.
10) The total amount should not exceed $4,000 in a single order.
11) The name and description should be mentioned as per the order specification (ex-specs for M1 should be different M1 Pro)
12) Amount in payments should be same with respect to total amount in orders table.

ENTITY RELATIONSHIP DIAGRAM:



DATA DICTIONARY:

For the benefit of programmers and others who needs to check the reference to the data items in a particular data model, a data dictionary is a combination of collected descriptions with the items. A data dictionary frequently serves as a centralized data repository.

The PROS AND CONS OF DATA DICTIONARY:

Some major contents in a dictionary can be different every time. Generally, these are components with various types of data provided and metadata types.

1. The names and definitions of the data objects
2. Properties of data element like data types, size, optionality, identifiers and indexes
3. Entity Relationship diagrams
4. Data that is referred
5. Business Rules
6. Missing data

APPLE STORE DATA DICTIONARY									
TABLE NAME	**ATTRIBUTE NAME**	**DESCRIPTION**	**CONTENTS**	**TYPE**	**FORMAT**	**RANGE**	**REQUIRED**	**PK OR FK**	**FK REFERENCED TABLE**
USERS	**User_id**	**Each user specific id**	**User ID**	**INT**	**9999**	**1000 - 9999**	**Y**	**PK**	
	Last_name	**last name of user**	**User Last Name**	**VARCHAR(50)**	**XXXXYYyyyy**				
	First_name	**first name of user**	**User First Name**	**VARCHAR(50)**	**sssddddhhSSSS**				
	Phone_Number	**phone number of user**	**User Phone Number**	**INT**	**999999999 99**				
	Email	**email of user**	**User Email Address**	**VARCHAR(60)**	**Xyyyncdc**				
	Address	**address of user**	**User Address**	**VARCHAR(60)**	**sdsdGJKK**				
ORDERS	**Order_id**	**Unique id of order**	**Order ID**	**INT**	**9999**	**1000 - 9999**	**Y**	**PK**	
	User_id	**Id of user who bought the order**	**User ID**	**INT**	**9999**	**1000 - 9999**	**Y**	**FK**	**USERS**
	Order_date	**Date of transaction**	**Ordered Date**	**DATE**	**DDMMYYYY**				

	Total_amount	**Total invoice amount**	**Payment Amount**	**NUMBER(10,2)**	**99999.99**				
ORDER ITEMS	**Shipment_id**	**Unique id of the order details**	**Shipment ID**	**INT**	**9999**	**1000 - 9999**	**Y**	**PK**	
	Order_id	**Unique id of the order**	**Order ID**	**INT**	**9999**	**1000 - 9999**	**Y**	**FK**	**USERS**
	Product_Id	**Unique product id**	**Product ID**	**INT**	**9999**	**1000 - 9999**	**Y**	**FK**	**PRODUCT**
	Quantity	**No of sold items**	**Quantity**	**INT**	**99999**	**10000-99999**			
PRODUCT_CATEGORY	**Category_Id**	**Unique category id**	**Category ID**	**INT**	**9999**	**1000 - 9999**	**Y**	**PK**	
	Category_Name	**Name of product**	**Name of Product**	**VARCHAR(60)**	**XXXyyyuu**				
	Description	**Desc of the product**	**Description**	**VARCHAR(70)**	**XXJJKKbbjjk**				
PRODUCT	**Product_Id**	**Unique product id**	**Product ID**	**INT**	**9999**	**1000 - 9999**	**Y**	**PK**	
	Product_Category	**product_category of the product**	**Product Category ID**	**INT**	**FFHassdj**		**Y**	**FK**	**PRODUCT_CATEGORY**
	Color	**Color of the product**	**Colour**	**VARCHAR(30)**	**HHJsadssf**				
	Product_Inventory	**Unique product inventory**	**Product Inventory**	**INT**	**9999**	**1000 - 9999**	**Y**	**FK**	**PRODUCT_INVENTORY**

	Price	**Price of the product**	**Price**	**NUMBER(10,2)**	**99999.99**				
PRODUCT_INVENTORY	**Inventory_Id**	**Unique inventory id**	**Inventory ID**	**INT**	**9999**	**1000-9999**	**Y**	**PK**	
	Quantity	**No of sold items**	**Quantity**	**INT**	**9999**	**1000-9999**			
PAYMENTS	**Payment_id**	**Unique payment id of each transaction**	**Payment Identification**	**INT**	**9999**	**1000-9999**	**Y**	**PK**	
	User_id	**Unique id of user**	**User Identification**	**INT**	**9999**	**1000-9999**	**Y**	**FK**	**USERS**
	Amount	**Total invoice amount**	**Amount**	**DECIMAL(10,2)**	**99999.99**				
	Provider	**Provider of the product**	**Providing service**	**VARCHAR(60)**	**FGHGHJK**				
	Status	**Stat of the transaction**	**Completion status**	**CHAR(40)**	**JBJ**				

QUERIES AND IT'S OPERATIONS FOR USERS TABLE:

Statements Explanation:

1) First, database is created in a database schema
2) Now, the tables are created in a database that is created. And named the table as Users.
3) Then, tables are filled with the related information that is required for the data base.
4) And the values are inserted.
5) Using the data dictionary, the information in the tables created.

Query:

```
CREATE TABLE `Users` (
  `User_id` int NOT NULL,
  `Last_name` char(50) COLLATE utf8mb4_general_ci NOT NULL,
  `First_name` char(50) COLLATE utf8mb4_general_ci NOT NULL,
  `Phone_number` bigint NOT NULL,
  `Email` varchar(60) COLLATE utf8mb4_general_ci NOT NULL,
  `Address` varchar(60) COLLATE utf8mb4_general_ci NOT NULL,
  PRIMARY KEY (`User_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Insert Statements:

```
INSERT INTO `Users` (`User_id`, `Last_name`, `First_name`, `Phone_number`, `Email`, `Address`) VALUES ('1', 'kunta', 'jivithesh', '9409997441', 'jivithesh@gmail.com', 'denton,texas'), ('2', 'gates', 'bill', '9409997448', 'bill@gmail.com', 'new york'), ('3', 'musk', 'elon', '9409997589', 'elon@spacex.com', 'california'), ('4', 'cameron', 'james', '9409997442', 'james@gmail.com', 'michigan'), ('5', 'jakkani', 'rashmi', '9409997896', 'rashmi@gmail.com', 'Texas'), ('6', 'gajjela', 'vyshnavi', '940999796', 'vyshnavi@gmail.com', 'Texas'), ('7', 'clinkz', 'anirudh', '2486838026', 'anirudh@gmail.com', 'Michigan'), ('8', 'wick', 'john', '9409785964', 'johnwick@gmail.com', 'Washington'), ('9', 'cruise', 'tom', '9402489874', 'tomcruise@gmail.com', 'Florida'), ('10', 'bond', 'james', '9409997896', 'rashmi@gmail.com', 'New York'), ('11', 'vergera', 'sofia', '9409999999', 'sofia@gmail.com', 'Florida'), ('12', 'gaga', 'lady', '7598157458', 'ladygaga@gmail.com', 'London'), ('13', 'bieber', 'justin', '9409997445', 'baby@gmail.com', 'New York'), ('14', 'timberlake', 'justin', '7409997445', 'timberlake@gmail.com', 'New York'), ('15', 'evans', 'chris', '9489997445', 'chrisevans@gmail.com', 'Florida'), ('16', 'johanssan', 'scarlet', '9404997445', 'scarlet@gmail.com', 'Florida'), ('17', 'smith', 'will', '9400997445', 'will@gmail.com', 'London'), ('18', 'Downey', 'Robert', '9400097445', 'downey@gmail.com', 'London'), ('19', 'lopez', 'jeniffer', '9477997445', 'jeniffer@gmail.com', 'vermont'), ('20', 'hathaway', 'anne', '9009997445', 'anne@gmail.com', 'New York'), ('21', 'reddy', 'monica', '9011117445', 'moni@gmail.com', 'Texas'), ('22', 'kadari', 'amulya', '9009991111', 'amulya@gmail.com', 'New York'), ('23', 'reddy', 'sathvica', '8009997445', 'sathvica@gmail.com', 'Texas'), ('24', 'iruku', 'srinivas', '9044997445', 'iruku@gmail.com', 'North Carolina'), ('25', 'kandregula', 'venketesh', '7709997445', 'venky@gmail.com', 'North Carolina')
```

Run SQL query/queries on table 5707moose.Users:

```
INSERT INTO `Users`(`User_id`, `Last_name`, `First_name`, `Phone_number`, `Email`, `Address`) VALUES
('11','vergera','sofia','9409999999','sofia@gmail.com','Florida'),
('12','gaga','lady','7598157458','ladygaga@gmail.com','London'),
('13','bieber','justin','9409997445','baby@gmail.com','New York'),
('14','timberlake','justin','7409997445','timberlake@gmail.com','New York'),
('15','evans','chris','9489997445','chrisevans@gmail.com','Florida'),
('16','johanssan','scarlet','9404997445','scarlet@gmail.com','Florida'),
('17','smith','will','9400997445','will@gmail.com','London'),
('18','Downey','Robert','9400097445','downey@gmail.com','London'),
('19','lopez','jeniffer','9477997445','jeniffer@gmail.com','vermont'),
('20','hathaway','anne','9009997445','anne@gmail.com','New York')
```

Result:

	User_id	Last_name	First_name	Phone_number	Email	Address
Edit Copy Delete	1	kunta	jivithesh	9409997441	jivithesh@gmail.com	denton,texas
Edit Copy Delete	2	gates	bill	9409997448	bill@gmail.com	new york
Edit Copy Delete	3	musk	elon	9409997589	elon@spacex.com	california
Edit Copy Delete	4	cameron	james	9409997442	james@gmail.com	michigan
Edit Copy Delete	5	jakkani	rashmi	9409997896	rashmi@gmail.com	Texas
Edit Copy Delete	6	gajjela	vyshnavi	940999796	vyshnavi@gmail.com	Texas
Edit Copy Delete	7	clinkz	anirudh	2486838026	anirudh@gmail.com	Michigan
Edit Copy Delete	8	wick	john	9409785964	johnwick@gmail.com	Washington
Edit Copy Delete	9	cruise	tom	9402489874	tomcruise@gmail.com	Florida
Edit Copy Delete	10	bond	james	9409997896	rashmi@gmail.com	New York
Edit Copy Delete	11	vergera	sofia	9409999999	sofia@gmail.com	Florida
Edit Copy Delete	12	gaga	lady	7598157458	ladygaga@gmail.com	London
Edit Copy Delete	13	bieber	justin	9409997445	baby@gmail.com	New York
Edit Copy Delete	14	timberlake	justin	7409997445	timberlake@gmail.com	New York
Edit Copy Delete	15	evans	chris	9489997445	chrisevans@gmail.com	Florida
Edit Copy Delete	16	johanssan	scarlet	9404997445	scarlet@gmail.com	Florida
Edit Copy Delete	17	smith	will	9400997445	will@gmail.com	London

Update Statement:

set address to texas in users where address =denton, texas.

Query:

UPDATE `Users` SET `Address` = 'texas' WHERE `Address`='denton, texas'

Show query box

5 rows affected. (Query took 0.0019 seconds.)

```
UPDATE `Users` SET `Address` = 'texas' WHERE `Address`='denton,texas';
```

[Edit inline] [Edit] [Create PHP code]

Data Retrieval:

The following are the conditions that are used to work on the goals to set SQL queries:

- To get all the data of user's data from the table the below is the select statement

Select Statements:

1. Show Texas users details.

Query:

```
SELECT * FROM `Users` where `Address`='Texas';
```

Result:

+ Options

	User_id	Last_name	First_name	Phone_number	Email	Address
Edit Copy Delete	5	jakkani	rashmi	9409997896	rashmi@gmail.com	Texas
Edit Copy Delete	6	gajjela	vyshnavi	940999796	vyshnavi@gmail.com	Texas
Edit Copy Delete	21	reddy	monica	9011117445	moni@gmail.com	Texas
Edit Copy Delete	23	reddy	sathvica	8009997445	sathvica@gmail.com	Texas

2. select user details whose phone_number starts with 8.?

Query:

```
SELECT * FROM `Users` where `Phone_number` LIKE '8%';
```

Result:

	User_id	Last_name	First_name	Phone_number	Email	Address
Edit Copy Delete	23	reddy	sathvica	8009997445	sathvica@gmail.com	Texas

QUERIES AND IT'S OPERATIONS FOR ORDERS TABLE:

Statements Explanation:

1) The database is created as 5707 moose.
2) Then, a table is created as orders table using the command ‘create table Orders’.
3) Now, the tables are inserted with the required information and the values are inserted.

Query:

```
CREATE TABLE `Orders` (
  `order_id` int NOT NULL,
  `User_id` int NOT NULL,
  `Order_date` date NOT NULL,
  `Total_amount` float NOT NULL,
  PRIMARY KEY (`order_id`),
  KEY `User_Id` (`User_id`),
  CONSTRAINT `User_Id` FOREIGN KEY (`User_id`) REFERENCES `Users` (`User_id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Insert Statements:

```
INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1500', '2', '2022-02-25', '2000')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1501', '7', '2022-02-15', '2700')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1502', '8', '2022-02-25', '500')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1503', '20', '2022-07-25', '3000')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1504', '15', '2022-01-25', '400')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1505', '11', '2022-05-25', '2850')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1506', '17', '2022-04-25', '1200')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1507', '8', '2022-02-15', '2700')
```

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1508', '4', '2021-01-25', '5598')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1509', '23', '2021-11-25', '5009')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1510', '1', '2021-10-25', '1400')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1511', '15', '2021-12-25', '3333')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1512', '21', '2021-04-14', '1256')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1513', '1', '2021-05-14', '1256')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1514', '14', '2021-06-14', '1506')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1515', '23', '2021-07-14', '2056')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1516', '24', '2020-08-14', '5600')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1517', '4', '2022-10-14', '3569')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1518', '5', '2021-05-14', '1290')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1519', '7', '2020-07-14', '1596')

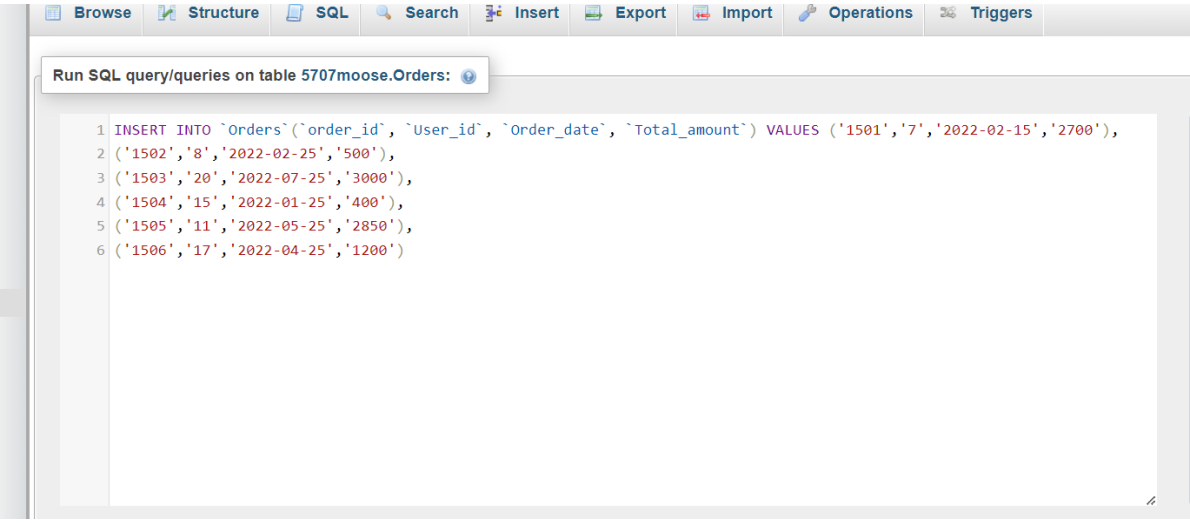
INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1520', '25', '2022-11-24', '700')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1521', '23', '2022-12-25', '6990')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1522', '4', '2021-04-25', '5980')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1523', '1', '2022-01-25', '1400')

INSERT INTO `Orders` (`order_id`, `User_id`, `Order_date`, `Total_amount`) VALUES ('1524', '15', '2022-02-25', '3300')



Result:

	order_id	User_id	Order_date	Total_amount
Edit Copy Delete	1500	2	2022-02-25	2000
Edit Copy Delete	1501	7	2022-02-15	2700
Edit Copy Delete	1502	8	2022-02-25	500
Edit Copy Delete	1503	20	2022-07-25	3000
Edit Copy Delete	1504	15	2022-01-25	400
Edit Copy Delete	1505	11	2022-05-25	2850
Edit Copy Delete	1506	17	2022-04-25	1200
Edit Copy Delete	1507	8	2022-02-15	2700
Edit Copy Delete	1508	4	2021-01-25	5598
Edit Copy Delete	1509	23	2021-11-25	5009
Edit Copy Delete	1510	1	2021-10-25	1400
Edit Copy Delete	1511	15	2021-12-25	3333
Edit Copy Delete	1512	21	2021-04-14	1256
Edit Copy Delete	1513	1	2021-05-14	1256

Update Statement:

Set amount=2500 for order_id 1500

Query:

```
UPDATE `Orders` SET `Total_amount` = '2500' WHERE `Orders`.`order_id` = 1500
```

1 row affected.

```
UPDATE `Orders` SET `Total_amount` = '2500' WHERE `Orders`.`order_id` = 1500;
```

[Edit inline] [Edit] [Create PHP code]

Data Retrieval:

The following are the conditions that are used to work on the goals to set SQL queries:

To retrieve the data of orders constraint, references are used and foreign key is ‘user id’, primary key is ‘order id.

Select Statements:

1. Select order details for which year is 2022 and total amount is more than 3000?

Query:

```
SELECT * FROM `Orders` WHERE YEAR(`Order_date`)=2022 AND `Total_amount`>3000;
```

Result:

+ Options

	order_id	User_id	Order_date	Total_amount
Edit Copy Delete	1517	4	2022-10-14	3569
Edit Copy Delete	1521	23	2022-12-25	6990
Edit Copy Delete	1524	15	2022-02-25	3300

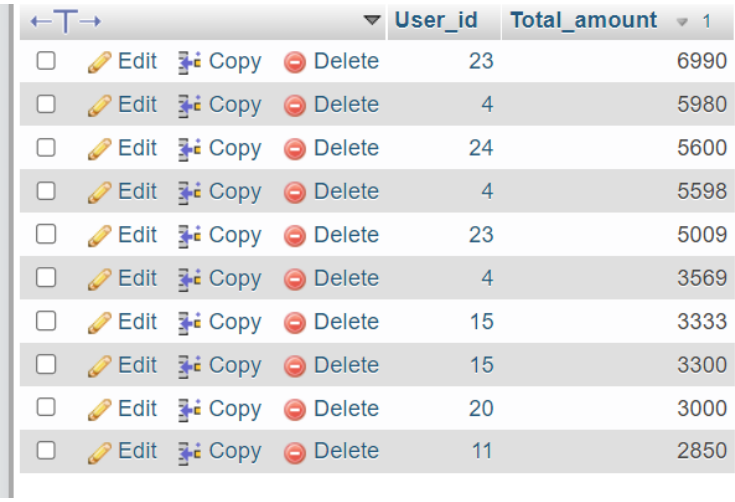
Check all *With selected:* Edit Copy Delete Export

2. Select top 10 user with highest order total ?

Query:

```
SELECT `User_id`, `Total_amount` FROM `Orders` ORDER BY `Total_amount` DESC LIMIT 10;
```

Result:



	User_id	Total_amount
Edit Copy Delete	23	6990
Edit Copy Delete	4	5980
Edit Copy Delete	24	5600
Edit Copy Delete	4	5598
Edit Copy Delete	23	5009
Edit Copy Delete	4	3569
Edit Copy Delete	15	3333
Edit Copy Delete	15	3300
Edit Copy Delete	20	3000
Edit Copy Delete	11	2850

QUERIES AND IT'S OPERATIONS FOR ORDER ITEMS TABLE:

Statements Explanation:

1) First, created a database named as 5707 moose
2) Then, table is created named as order items using command 'create table order_items'
3) Now, the required data is added and information is present in the data.

Query:

```
CREATE TABLE `Order Items` (
  `shipment_id` int NOT NULL,
  `Order_id` int NOT NULL,
  `Product_Id` int NOT NULL,
  `Quantity` int NOT NULL,
  PRIMARY KEY (`shipment_id`),
  KEY `Order_id` (`Order_id`),
  KEY `Product_Id` (`Product_Id`),
  CONSTRAINT `Order_id` FOREIGN KEY (`Order_id`) REFERENCES `Orders` (`order_id`) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT `Product_Id` FOREIGN KEY (`Product_Id`) REFERENCES `Product` (`Product_Id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Insert Statements:

INSERT INTO `Order Items` (`shipment_id`, `Order_id`, `Product_Id`, `Quantity`) VALUES ('2001', '1569', '1504', '1'), ('2003', '1502', '4', '1'), ('2004', '1503', '5', '1'), ('2005', '1504', '8', '2'), ('2006', '1504', '11', '3'), ('2007', '1506', '9', '1'), ('2008', '1507', '10', '5'), ('2009', '1508', '11', '1'), ('2010', '1510', '13', '4'), ('2011', '1510', '15', '3'), ('2012', '1512', '7', '2'), ('2013', '1513', '5', '1'), ('2015', '1513', '3', '2'), ('2016', '1502', '3', '2'), ('2017', '1513', '11', '1'), ('2018', '1513', '2', '1'), ('2019', '1503', '12', '1'), ('2020', '1514', '2', '1'), ('2021', '1515', '6', '1'), ('2022', '1507', '11', '2'), ('2023', '1506', '11', '1'), ('2024', '1502', '11', '4'), ('2025', '1508', '11', '2'), ('2026', '1520', '11', '5')

```
INSERT INTO `Order Items`(`shipment_id`, `Order_id`, `Product_Id`, `Quantity`) VALUES ('2002','1502','3','2'),
('2002','1502','3','2'),
('2003','1502','4','1'),
('2004','1503','5','1'),
('2005','1504','8','2'),
('2006','1504','1','3'),
('2007','1506','9','1'),
('2008','1507','10','5'),
('2009','1508','11','1'),
('2010','1510','13','4'),
('2011','1510','15','3'),
('2012','1512','7','2'),
('2013','1513','5','1')
```

Result:

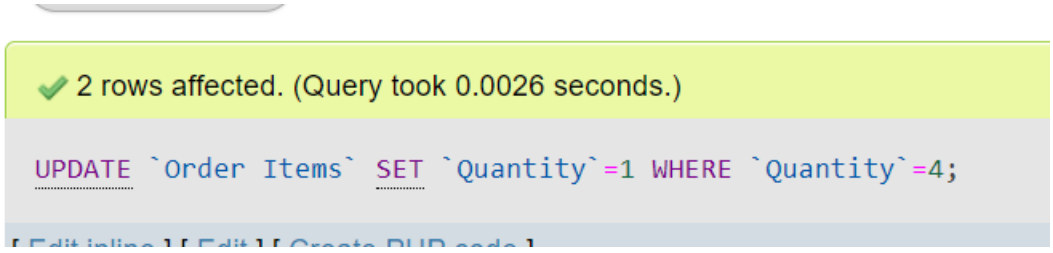
	shipment_id	Order_id	Product_Id	Quantity
Edit Copy Delete	2001	1569	1504	1
Edit Copy Delete	2003	1502	4	1
Edit Copy Delete	2004	1503	5	1
Edit Copy Delete	2005	1504	8	2
Edit Copy Delete	2006	1504	11	3
Edit Copy Delete	2007	1506	9	1
Edit Copy Delete	2008	1507	10	5
Edit Copy Delete	2009	1508	11	1
Edit Copy Delete	2010	1510	13	4
Edit Copy Delete	2011	1510	15	3
Edit Copy Delete	2012	1512	7	2
Edit Copy Delete	2013	1513	5	1
Edit Copy Delete	2015	1513	3	2
Edit Copy Delete	2016	1502	3	2

Update Statement:

Set quantity =1 where quantity = 4

Query:

```
UPDATE `Orders Items` SET `Quantity` = 1 WHERE 'Quantity'=4;
```



Data Retrieval:

The following are the conditions that are used to work on the goals to set SQL queries:

1) To retrieve the data of orders items table is used.

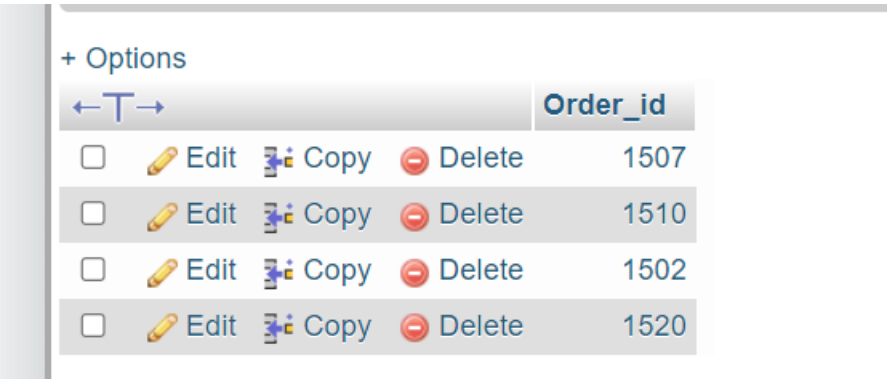
2) For constraint, order id the foreign key as order id on delete constraint on update restrict is used.
3) For constraint, product id the foreign key as product id on delete constraint on update restrict is used where as the primary key is shipment id.

Select Statements:

1: Select order id where quantity is more than 3?

Query: SELECT `Order_id` FROM `Order Items` WHERE `Quantity`>3;

Result:

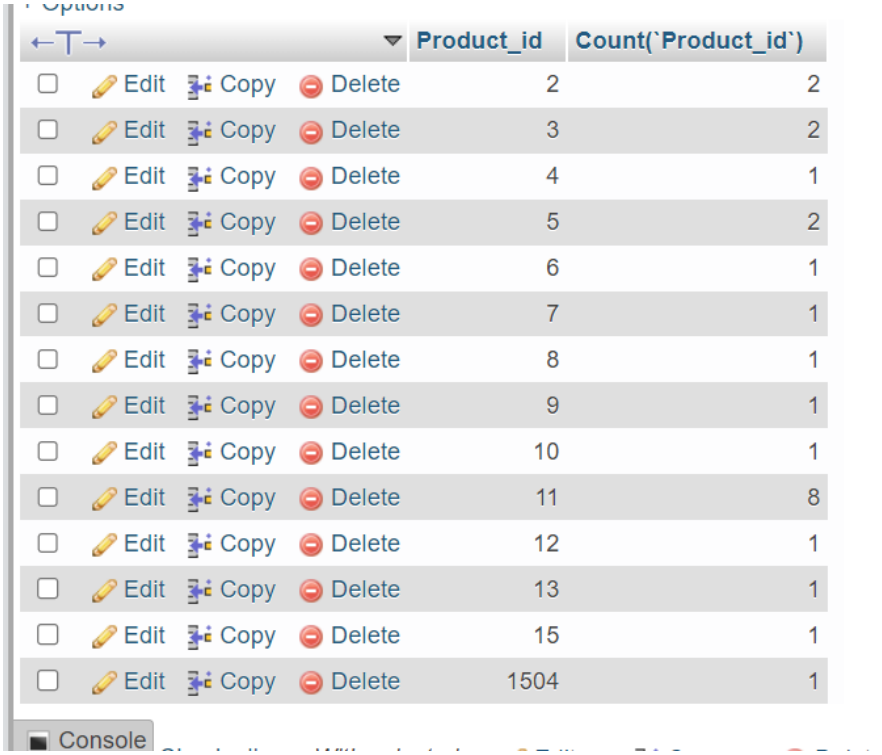


2. How many times each product is ordered?

Query:

SELECT `Product_id`,Count(`Product_id`) FROM `Order Items` GROUP BY `Product_id`;

Result:



	Product_id	Count(`Product_id`)
Edit Copy Delete	2	2
Edit Copy Delete	3	2
Edit Copy Delete	4	1
Edit Copy Delete	5	2
Edit Copy Delete	6	1
Edit Copy Delete	7	1
Edit Copy Delete	8	1
Edit Copy Delete	9	1
Edit Copy Delete	10	1
Edit Copy Delete	11	8
Edit Copy Delete	12	1
Edit Copy Delete	13	1
Edit Copy Delete	15	1
Edit Copy Delete	1504	1

Console

QUERIES AND IT'S OPERATIONS FOR PAYMENTS TABLE:

Statements Explanation:

1) First, created a database named as 5707 moose
2) Then, table is created named as payments using command 'create table payments'
3) Now, the required data is added and information is present in the data.

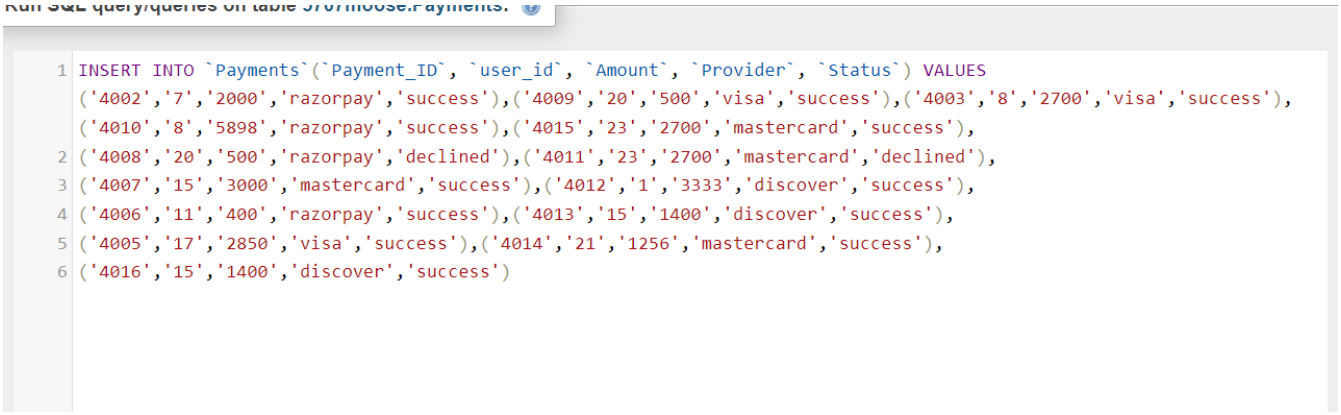
Query:

```
CREATE TABLE `Payments` (
  `Payment_ID` int NOT NULL,
  `user_id` int NOT NULL,
  `Amount` decimal(10,2) NOT NULL,
  `Provider` varchar(60) COLLATE utf8mb4_general_ci NOT NULL,
  `Status` char(40) COLLATE utf8mb4_general_ci NOT NULL,
  PRIMARY KEY (`Payment_ID`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `Payments_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `Users`
(`User_id`) ON DELETE RESTRICT ON UPDATE RESTRICT
```

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci

Insert Statements:

INSERT INTO `Payments` (`Payment_ID`, `user_id`, `Amount`, `Provider`, `Status`) VALUES ('4001', '1', '1000.00', 'Amex', 'Success'), ('4002', '7', '2000.00', 'razorpay', 'success'), ('4003', '8', '2700.00', 'visa', 'success'), ('4005', '17', '2850.00', 'visa', 'success'), ('4006', '11', '400.00', 'razorpay', 'success'), ('4007', '15', '3000.00', 'mastercard', 'success'), ('4008', '20', '500.00', 'razorpay', 'declined'), ('4009', '20', '500.00', 'visa', 'success'), ('4010', '8', '5898.00', 'razorpay', 'success'), ('4011', '23', '2700.00', 'mastercard', 'declined'), ('4012', '1', '3333.00', 'discover', 'success'), ('4013', '15', '1400.00', 'discover', 'success'), ('4014', '21', '1256.00', 'mastercard', 'success'), ('4015', '23', '2700.00', 'mastercard', 'success'), ('4016', '15', '1400.00', 'discover', 'success'), ('4017', '2', '1561.00', 'Visa', 'Success'), ('4018', '3', '2561.00', 'discover', 'Declined'), ('4019', '4', '3561.00', 'amex', 'declined'), ('4020', '5', '4561.00', 'Visa', 'Success'), ('4021', '16', '1562.00', 'mastercard', 'Success'), ('4022', '20', '1761.00', 'mastercard', 'Declined'), ('4023', '21', '1961.00', 'Discover', 'Success'), ('4024', '22', '1701.00', 'mastercard', 'Success'), ('4025', '18', '6524.00', 'mastercard', 'Success'), ('4026', '16', '8547.00', 'Visa', 'Success')



```
1 INSERT INTO `Payments`(`Payment_ID`, `user_id`, `Amount`, `Provider`, `Status`) VALUES
  ('4002','7','2000','razorpay','success'),('4009','20','500','visa','success'),('4003','8','2700','visa','success'),
  ('4010','8','5898','razorpay','success'),('4015','23','2700','mastercard','success'),
2 ('4008','20','500','razorpay','declined'),('4011','23','2700','mastercard','declined'),
3 ('4007','15','3000','mastercard','success'),('4012','1','3333','discover','success'),
4 ('4006','11','400','razorpay','success'),('4013','15','1400','discover','success'),
5 ('4005','17','2850','visa','success'),('4014','21','1256','mastercard','success'),
6 ('4016','15','1400','discover','success')
```

Result:

+ Options

	Payment_ID	user_id	Amount	Provider	Status
Edit Copy Delete	4001	1	1000.00	Amex	Success
Edit Copy Delete	4002	7	2000.00	razorpay	success
Edit Copy Delete	4003	8	2700.00	visa	success
Edit Copy Delete	4005	17	2850.00	visa	success
Edit Copy Delete	4006	11	400.00	razorpay	success
Edit Copy Delete	4007	15	3000.00	mastercard	success
Edit Copy Delete	4008	20	500.00	razorpay	declined
Edit Copy Delete	4009	20	500.00	visa	success
Edit Copy Delete	4010	8	5898.00	razorpay	success
Edit Copy Delete	4011	23	2700.00	mastercard	declined
Edit Copy Delete	4012	1	3333.00	discover	success
Edit Copy Delete	4013	15	1400.00	discover	success
Edit Copy Delete	4014	21	1256.00	mastercard	success
Edit Copy Delete	4015	23	2700.00	mastercard	success
Edit Copy Delete	4016	15	1400.00	discover	success

Update Statement:

Update status=success for payment id 4011

Query:

```
UPDATE `Payments` SET `Status` = 'success' WHERE `Payments`.`Payment_ID` = 4011
```

1 row affected.

```
UPDATE `Payments` SET `Status` = 'success' WHERE `Payments`.`Payment_ID` = 4011;
```

[Edit inline] [Edit] [Create PHP code]

Data Retrieval:

The following are the conditions that are used to work on the goals to set SQL queries:

1) To retrieve the data of payments table is used.
2) For constraint, Payments_ibfk_1 the foreign key as user id and references as user id on delete constraint on update restrict is used and primary key is payment id.

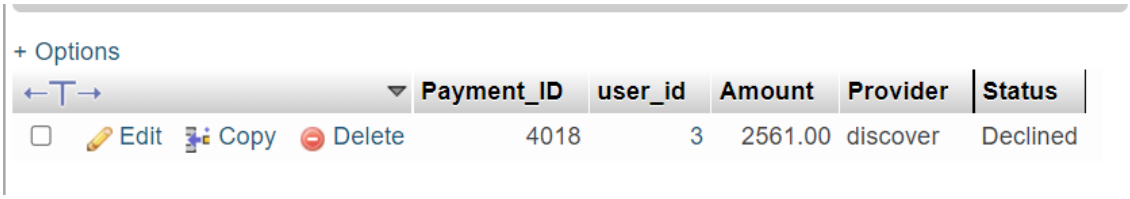
Select Statements:

1. Select payment details where provider discover and status is declined?

Query:

```
SELECT * FROM `Payments` WHERE `Status`='Declined' and `Provider`='discover';
```

Result:

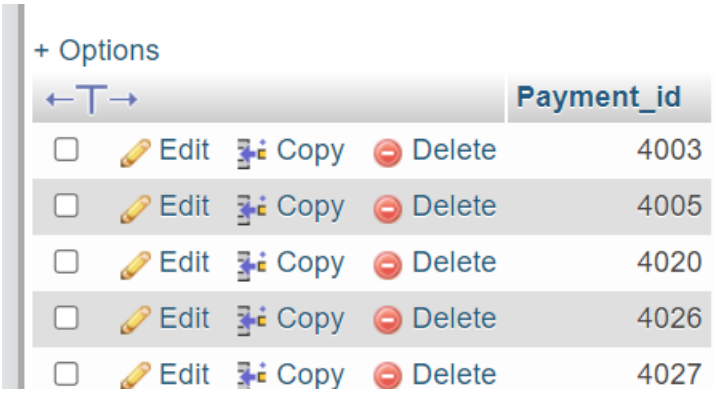


2. select payment id where amount is more than 2000 and provider is visa?

Query:

```
SELECT `Payment_id` FROM `Payments` WHERE `Amount`>2000 and `Provider`='visa';
```

Result:



QUERIES AND IT'S OPERATIONS FOR PRODUCT TABLE:

Statements Explanation:

1) First, created a database named as 5707 moose
2) Then, table is created named as product using command 'create table product'
3) Now, the required data is added and information is present in the data.

Query:

```
CREATE TABLE `Product` (
  `Product_Id` int NOT NULL,
  `Product_name` text COLLATE utf8mb4_general_ci NOT NULL,
```

```
  `Product_Category` int NOT NULL,
  `Color` char(30) COLLATE utf8mb4_general_ci NOT NULL,
  `Product_Inventory` int NOT NULL,
  `Price` decimal(10,2) NOT NULL,
  PRIMARY KEY (`Product_Id`),
  KEY `Product_inventory` (`Product_Inventory`),
  KEY `Product_Category` (`Product_Category`),
  CONSTRAINT `Product_Category` FOREIGN KEY (`Product_Category`) REFERENCES `Product_Category` (`Category_Id`) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT `Product_inventory` FOREIGN KEY (`Product_Inventory`) REFERENCES `Product_Inventory` (`Inventory_id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Insert Statements:

```
INSERT INTO `Product` (`Product_Id`, `Product_name`, `Product_Category`, `Color`, `Product_Inventory`, `Price`) VALUES ('2', 'Iphone 13 Pro', '1', 'Red', '1002', '1000.00'), ('3', 'Iphone 13 Pro', '1', 'Graphite', '1002', '1000.00'), ('4', 'Iphone 13 Pro max', '1', 'White', '1004', '1200.00'), ('5', 'Iphone 13 Pro max', '1', 'Gold', '1007', '1200.00'), ('6', 'Iphone 13 pro max', '1', 'Sierra Blue', '1006', '1200.00'), ('7', 'Ipad pro ', '2', 'Sierra blue', '1007', '999.00'), ('8', 'Ipad 8th gen ', '2', 'White', '1400', '499.00'), ('9', 'Ipad air ', '2', 'Gold', '1403', '699.00'), ('10', 'Ipad mini ', '2', 'Graphite', '4004', '599.00'), ('11', 'Macbook air ', '3', 'Grey', '1703', '1199.00'), ('12', 'Macbook pro ', '3', 'Gold', '1804', '1999.00'), ('13', 'Apple Charger', '4', 'white', '1002', '20.00'), ('14', 'Phone Case', '4', 'clear', '1804', '99.00'), ('15', 'Wireless charger', '4', 'Gold', '1804', '99.00'), ('20', 'Iphone 13 Pro', '1', 'Red', '1002', '1000.00'), ('25', 'Iphone 13 Pro', '1', 'Black', '1002', '1000.00'), ('40', 'Iphone 13 Pro max', '1', 'Gold', '1004', '1200.00'), ('50', 'Iphone 13 Pro max', '1', 'White', '1007', '1200.00'), ('60', 'Iphone 13 pro max', '1', 'White', '1006', '1200.00'), ('70', 'Ipad pro ', '2', 'Sierra blue', '1007', '999.00'), ('80', 'Ipad 8th gen ', '2', 'Sierra blue', '1400', '499.00'), ('90', 'Ipad air ', '2', 'Green', '1403', '699.00'), ('100', 'Ipad mini ', '2', 'Green', '4004', '599.00'), ('110', 'Macbook air ', '3', 'Pink', '1703', '1199.00'), ('120', 'Macbook pro ', '3', 'Gold', '1804', '1999.00')
```

```
INSERT INTO `Product`(`Product_Id`, `Product_name`, `Product_Category`, `Color`, `Product_Inventory`, `Price`) VALUES
('2','Iphone 13 Pro','1','Red','1002','1000'),
('3','Iphone 13 Pro','1','Graphite','1002','1000'),
('4','Iphone 13 Pro max','1','White','1004','1200'),
('5','Iphone 13 Pro max','1','Gold','1005','1200'),
('6','Iphone 13 pro max','1','Sierra Blue','1006','1200'),
('7','Ipad pro ','2','Sierra blue','1007','999'),
('8','Ipad 8th gen ','2','White','1400','499'),
('9','Ipad air ','2','Gold','1403','699'),
('10','Ipad mini ','2','Graphite','4004','599'),
('11','Macbook air ','3','Grey','1703','1199'),
('12','Macbook pro ','3','Gold','1804','1999'),
('13','Apple Charger','4','white','1002','20'),
('14','Phone Case','4','clear','1804','99'),
('15','Wireless charger','4','Gold','1804','99')
```

Result:

+ Options

	Product_Id	Product_name	Product_Category	Color	Product_Inventory	Price
Edit Copy Delete	2	Iphone 13 Pro	1	Red	1002	1000.00
Edit Copy Delete	3	Iphone 13 Pro	1	Graphite	1002	1000.00
Edit Copy Delete	4	Iphone 13 Pro max	1	White	1004	1200.00
Edit Copy Delete	5	Iphone 13 Pro max	1	Gold	1007	1200.00
Edit Copy Delete	6	Iphone 13 pro max	1	Sierra Blue	1006	1200.00
Edit Copy Delete	7	Ipad pro	2	Sierra blue	1007	999.00
Edit Copy Delete	8	Ipad 8th gen	2	White	1400	499.00
Edit Copy Delete	9	Ipad air	2	Gold	1403	699.00
Edit Copy Delete	10	Ipad mini	2	Graphite	4004	599.00
Edit Copy Delete	11	Macbook air	3	Grey	1703	1199.00
Edit Copy Delete	12	Macbook pro	3	Gold	1804	1999.00
Edit Copy Delete	13	Apple Charger	4	white	1002	20.00
Edit Copy Delete	14	Phone Case	4	clear	1804	99.00
Edit Copy Delete	15	Wireless charger	4	Gold	1804	99.00
Edit Copy Delete	1504	Iphone 13 Pro	1	sierra blue	1001	1000.00

Update Statement:

update color of products from black to Graphite

Query:

UPDATE `Product` SET `Color`='[value-4]' WHERE `Color`='Black'

1 row affected. (Query took 0.0113 seconds.)

```
UPDATE `Product` SET `Color`='[value-4]' WHERE `Color`='Black';
```

[Edit inline] [Edit] [Create PHP code]

Data Retrieval:

The following are the conditions that are used to work on the goals to set SQL queries:

1) To retrieve the data of product table is used.
2) For constraint, product category the foreign key as product category and references as product id on delete constraint on update restrict is used.
3) For constraint, product inventory the foreign key as product inventory and references as inventory id on delete constraint on update restrict is used.

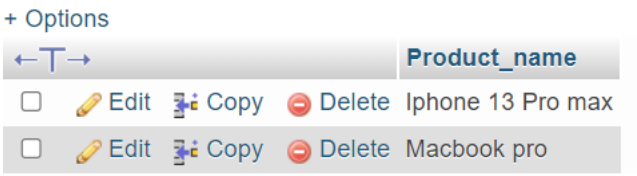
Select Statements:

1. Select product whose colour is gold and price more than 1000

Query:

```
SELECT `Product_name` FROM `Product` WHERE `Color`='Gold' and `Price` >1000;
```

Result:

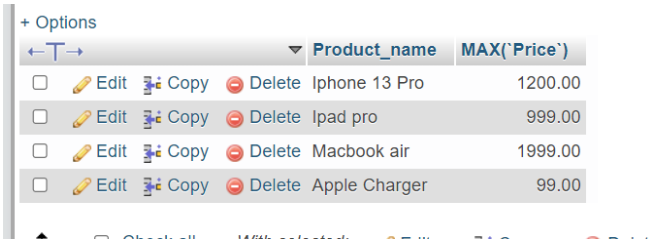


2. select costly product in each category.

Query:

```
SELECT `Product_name`,MAX(`Price`) FROM `Product` GROUP BY `Product_category`;
```

Result:



QUERIES AND IT'S OPERATIONS FOR PRODUCT CATEGORY TABLE:

Statements Explanation:

1) First, created a database named as 5707 moose
2) Then, table is created named as product_category using command 'create table product_category'

3) Now, the required data is added and information is present in the data.

Query:

```
CREATE TABLE `Product_Category` (
  `Category_Id` int NOT NULL,
  `Name` char(60) COLLATE utf8mb4_general_ci NOT NULL,
  `Description` char(70) COLLATE utf8mb4_general_ci NOT NULL,
  PRIMARY KEY (`Category_Id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Insert Statements:

INSERT INTO `Product_Category` (`Category_Id`, `Name`, `Description`) VALUES ('1', 'Iphone', 'Mobile Phones'), ('2', 'Ipad', 'Tablets'), ('3', 'Mac', 'Computer'), ('4', 'accessories', 'accessories for different products'), ('5', 'watch', 'apple watch'), ('6', 'tv', 'apple tv')

Result:

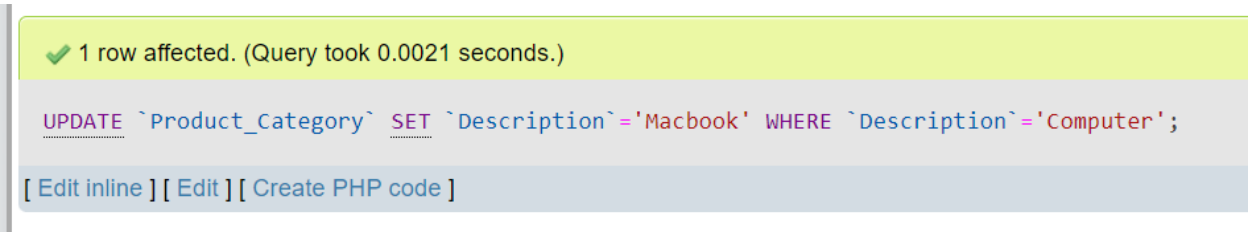
	Category_Id	Name	Description
Edit Copy Delete	1	Iphone	Mobile Phones
Edit Copy Delete	2	Ipad	Tablets
Edit Copy Delete	3	Mac	Computer
Edit Copy Delete	4	accessories	accessories for different products
Edit Copy Delete	5	watch	apple watch
Edit Copy Delete	6	tv	apple tv

Update Statement:

update description to MacBook where description is equal to computer

Query:

UPDATE `Product_Category` SET `Description`='Macbook' WHERE `Description`='Computer'



Data Retrieval:

The following are the conditions that are used to work on the goals to set SQL queries:

1) To retrieve the data of product category table is used.
2) To retrieve the category id, category name and its description are displayed on the result panel.

Select Statements:

1. How many categories are there.?

Query:

SELECT COUNT(*) FROM `Product_Category`;

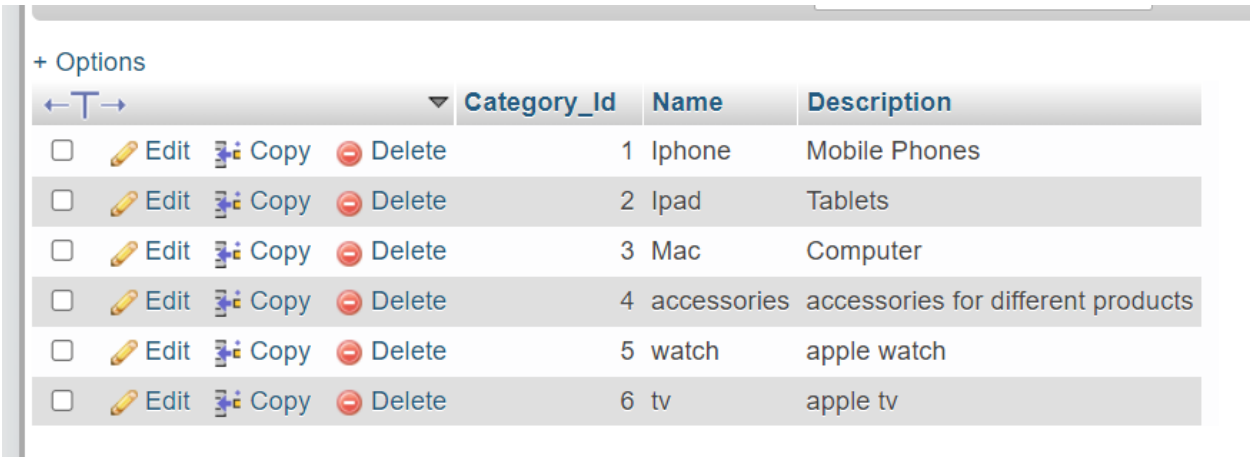
Result:

2. Display the details of product category?

Query:

SELECT * FROM `Product_Category`;

Result:



+ Options

	Category_Id	Name	Description
Edit Copy Delete	1	Iphone	Mobile Phones
Edit Copy Delete	2	Ipad	Tablets
Edit Copy Delete	3	Mac	Computer
Edit Copy Delete	4	accessories	accessories for different products
Edit Copy Delete	5	watch	apple watch
Edit Copy Delete	6	tv	apple tv

QUERIES AND IT'S OPERATIONS FOR PRODUCT INVENTORY TABLE:

Statements Explanation:

1) First, created a database named as 5707 moose
2) Then, table is created named as product inventory using command 'create table product _inventory'
3) Now, the required data is added and information is present in the data.

Query:

```
CREATE TABLE `Product_Inventory` (
  `Inventory_id` int NOT NULL,
  `Quantity` int NOT NULL,
  PRIMARY KEY (`Inventory_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Insert Statements:

```
INSERT INTO `Product_Inventory` (`Inventory_id`, `Quantity`) VALUES ('1001', '20'), ('1002', '30'), ('1003', '25'), ('1004', '30'), ('1006', '25'), ('1007', '30'), ('1008', '1'), ('1009', '5'), ('1010', '8'), ('1011', '8'), ('1012', '5'), ('1013', '7'), ('1014', '1'), ('1015', '2'), ('1016', '3'), ('1017', '4'), ('1018', '1'), ('1019', '4'), ('1020', '1'), ('1021', '1'), ('1022', '1'), ('1033', '1'), ('1400', '10'), ('1403', '25'), ('1503', '25')
```

```
INSERT INTO `Product_Inventory`(`Inventory_id`, `Quantity`) VALUES ('1006','25'),('1007','30'),('1503','25'),
('4004','40'),('1703','15'),('1804','37'),('1403','25'),('1400','10')
```

Result:

Options

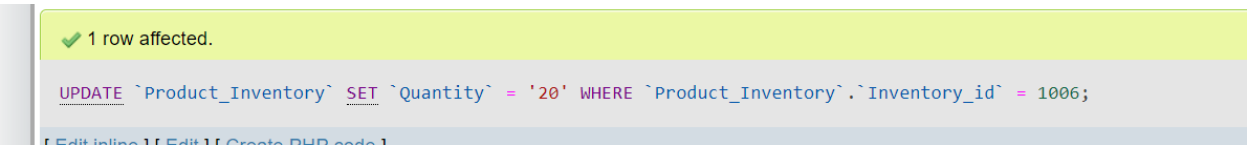
	Inventory_id	Quantity
Edit Copy Delete	1001	20
Edit Copy Delete	1002	30
Edit Copy Delete	1003	25
Edit Copy Delete	1004	30
Edit Copy Delete	1006	25
Edit Copy Delete	1007	30
Edit Copy Delete	1400	10
Edit Copy Delete	1403	25
Edit Copy Delete	1503	25
Edit Copy Delete	1703	15
Edit Copy Delete	1804	37
Edit Copy Delete	4004	40

Update Statement:

Update quantity where inventory_id is equal to 1006

Query:

```
UPDATE `Product_Inventory` SET `Quantity` = '20' WHERE `Product_Inventory`.`Inventory_id` = 1006
```



Data Retrieval:

The following are the conditions that are used to work on the goals to set SQL queries:

1) To retrieve the data of product inventory table is used.
2) For constraints, inventory id is the primary key. To retrieve and analyse the data this is performed.

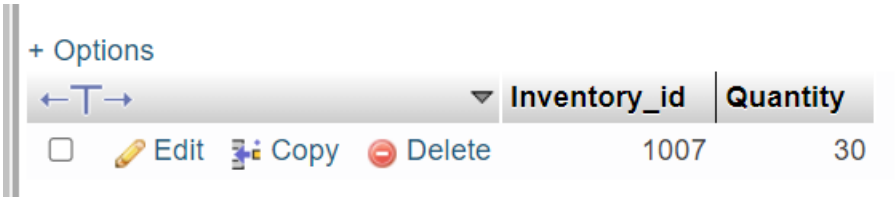
Select Statements:

1. select inventory details for inventory id 1007

Query:

```
SELECT * FROM `Product_Inventory` WHERE `Inventory_id`=1007;
```

Result:



2. select product with max inventory.

Query:

```
SELECT `Inventory_id`,max(`Quantity`) FROM `Product_Inventory`;
```

Result:

Data Retrieval and Sample Reports:

Here we are displaying some sample reports with queries and their results:

1. Show user details for whom payment got failed?

Explanation: In this query we are finding the user details for whom payment got failed

Query:

```
SELECT u.`First_name`,u.`Last_name`,p.`status` from Users u join Payments p ON u.`user_id`=p.`user_id` where `status`='declined';
```

Result:

+ Options

First_name	Last_name	status
anne	hathaway	declined
sathvica	reddy	declined
elon	musk	Declined
james	cameron	declined
anne	hathaway	Declined

2. Show category_name for product with id 4 and display the stock details?

Explanation: In this query we are finding the category name of the product with id 4 and displayed the details of the stock

Query:

```
SELECT c.`name`,i.`Quantity` from Product p join Product_Category c on p.`Product_Category`=c.`Category_id` join Product_Inventory i on p.`Product_inventory`=i.`inventory_id` where p.`Product_id`=4;
```

Result:

+ Options

name	Quantity
Iphone	30

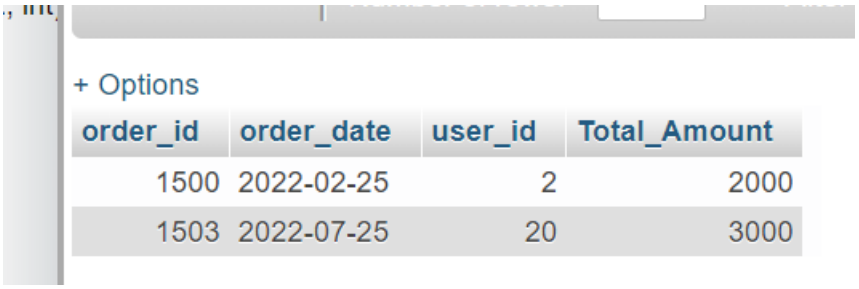
3. Show order details which were placed by user in New York and total amount is more than 2000 in the year 2022

Explanation: In this query we are finding the order details of the New York that are placed and the total amount in the year 2022 is more than 2000

Query:

```
select o.`order_id`,o.`order_date`,o.`user_id`,o.`Total_Amount` from Users u join Orders o on u.`user_id`=o.`user_id` where Address='New York' and year(order_date)=2022;
```

Result:



+ Options

order_id	order_date	user_id	Total_Amount
1500	2022-02-25	2	2000
1503	2022-07-25	20	3000

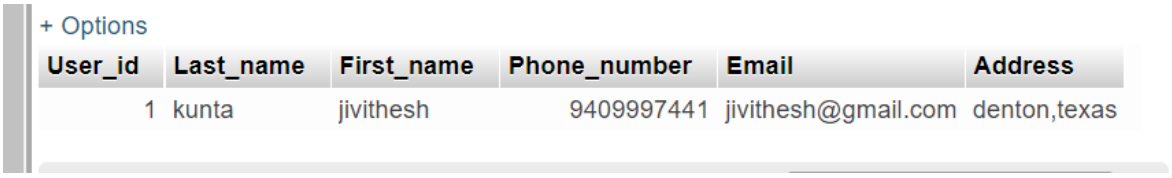
4. Show details of user who bought I phone 13 pro blue colour mobile phone.

Explanation: In this we are finding the user details who bought I phone 13 pro blue color mobile phone.

Query:

```
SELECT u.* from Users u join Orders o on u.`user_id`=o.`user_id` JOIN `Order Items` ot on o.`order_id`=ot.`order_id` JOIN Product p on ot.`product_id`=p.`product_id` WHERE p.Product_name='iphone 13 pro' and color='Sierra Blue';
```

Result:



+ Options

User_id	Last_name	First_name	Phone_number	Email	Address
1	kunta	jivithesh	9409997441	jivithesh@gmail.com	denton,texas

5. show order details for I phone where price is more than 1000

Explanation: In this we are finding the order details of the I phone with price more than 100

Query:

```
SELECT ot.* FROM `Order Items` ot join Product p on ot.`product_id`=p.`product_id` join Product_Category c on p.`Product_Category`=c.`Category_id` where name='iphone';
```

Result:

+ Options

shipment_id	Order_id	Product_Id	Quantity
2018	1513	2	1
2020	1514	2	1
2015	1513	3	2
2016	1502	3	2
2003	1502	4	1
2004	1503	5	1
2013	1513	5	1
2021	1515	6	1
2001	1569	1504	1

Conclusion:

Hence, we can say that Apple store can change the landscape of consumer electronics retailers and influenced other technological companies. This Apple store database can be used for performing day to day transactions in a store which specializes in apple products. This is will decrease the difficulty in storing all the information of products and tracking the customer orders and simultaneously they can plan on ordering new stock for the store so that they keep the customers satisfied by always keeping products which the customers want.