# SQL – Class Exercises Solutions

**Instructions:**

This document contains all the practice exercises solutions conducted during the session. This is based on the sample case study described in the background section. The case study is a simplified version to real life scenarios. All names used in these exercises are indicative and do not intend or refer to any person or brand in real world.

**Background:**

MyShop is a retail ecommerce application built for customers to register themselves, add their preferred products to their shopping cart and later place orders. The MyShop data architect has already designed the data model for us to work on the same. Here the solutions for most of the exercises we work in the class.

**Exercise Solutions:**

1. Creation of Database.
   a. Using SQL create database statement, please create the database of MyShop using MYSQL workbench.
   b. Once created, check what are the database objects you see in the navigation pane.

Solution: This will be done in MySQL Workbench. Alternatively, you can use the script below to create the database.

```
CREATE DATABASE MyShop
```

2. Creation of table
   a. Let's create our first table "Customers" as per the design specification given in Appendix-1.

Solution: Appendix-1 gives the script for creating the customers table.

3. INSERT/UPDATE/Delete :
   a. Let's insert 3 customers in our customers table.
      Solution:

```
INSERT INTO Customers(customer_id,
cust_first_name,
cust_middle_name,
cust_last_name,
cust_phone,
cust_address_1,
cust_pincode,
cust_state,
cust_country,
cust_dateOfBirth)
VALUES ("00001","Ramesh","R","Misaara",
"324342342",
"20A , Dip Socty Mumbai",
"200002",
"Maharashtra",
"India",
"1943-01-01");
```

b. Customer2 whose last_name is “Misra” is misspelled. How will we update the record?
   Solution:
   ```
   Update customers
   set cust_last_name = "Misra"
   where cust_last_name = "Misaara";
   ```

c. Customer3 details are totally wrong. Name entered is “Rajiv”. Well we need to delete this customer.
   Solution:
   ```
   Delete from customers where first_name= “Rajiv”
   ```

4. Creation of the tables and load sample data
   a. We need to create all the tables in the MyShop database. Try creating a SQL Script and create all the tables. You can take help of the sample scripts and create others.
      Solution:
      `Appendix-1 gives the script for all the tables in our MyShop database.`
   b. Let’s load sample data in all the tables.

      Solution:
      ```
      You have already created sample insert script. Leveraging the
      same you can create the sample test data. Points to note when
      you have to create sample test data creation insert tables
      scripts
         a. If there is an auto-increment column, then do not put a
            value for the same
         b. If there are foreign keys defined, sample data needs to
            be inserted accordingly
         c. If there are triggers defined, you should ensure all
            referential integrity is maintained and masters data are
            inserted before any transactional data
         d. Only if you want the default value to be applied, do not
            enter a value for such columns
         e. Check the query conditions for which sample test data
            needs to be created and accordingly create/modify the
            data creation scripts.
      ```

5. Data Manipulation - Select
   a. Find all the list of products.
      Solution:
      ```
      Select * from products;
      ```
   b. Get the product name, product_cost of those products which are out of stock.
      Solution:
      ```
      Select prod_name,prod_price_unit from products where
      prod_status="AVL";
      ```
   c. Get the products which are either out of stock or Pending arrival from company. (Values - “NOSTOCK”, “AWTARRIVAL”)

```
Select  prod_name,prod_price_unit  from  products  where
prod_status in ("NOSTOCK","AWTARRIVAL");
```

d. Get the customer list who have done purchases in last 6 months (180 days) roughly.
   Solution:
```
select ct.customer_id, cust_first_name
from customers ct, orders od
where ct.customer_id = od.customer_id
and datediff(CURRENT_DATE(), od.order_date) <= 180;
```

e. Get all those customers who have purchased products costing above 1000 INR in that last 3 months.
```
select  ct.customer_id,  ct.cust_first_name  from  customers
ct, orders od, products pd, product_orders po
where ct.customer_id = od.customer_id
and od.order_no = po.order_no
and pd.prod_id = po.prod_id
and po.price_per_unit > 1000
and
order_date >= date_sub(current_date(),INTERVAL,3,MONTH);
```

f. Identify all those customers who have returned their orders for refund in the last 3 months.
```
select customer_id, order_no
from orders
where order_status = "REFUND"
and (Month(current_date()) - Month(order_date)) = 3
```

g. Get all those products in the ascending order of the cost whose cost is above 150 but below 500
```
Solution:
select prod_id, prod_name, prod_price_unit
from products
where prod_price_unit between 150 and 500
order by prod_price_unit asc
```

6. Data Manipulation - Joins

a. We need to get list of all the customers with the total count of products they have purchased.
```
select ct.customer_id, count(po.prod_id)
from customers ct left outer join orders od
on ct.customer_id = od.customer_id
left outer join product_orders po
on od.order_no = po.order_no
```

b. From all the orders placed, there is a need to identify the total count of orders against the complete list of products.

```
select pd.prod_id, count(po.order_no)
from product_orders po right outer join products pd
on po.prod_id = pd.prod_id;
```

c. We need to get the top 10 customers, the total value of their orders they have purchased in the last 3 months. A customer comes in top 10 by their order value.
/* This query will return us the total value of their orders and not the value for last 3 months */

```
Select     ct.customer_id,     sum(po.price_per_unit     *
po.order_qty) as 'total_value'
from customers ct left outer join orders od
on ct.customer_id = od.customer_id
left outer join product_orders po
on od.order_no = po.order_no
group by ct.customer_id, od.order_no
order by total_value
Limit 10
```

If we have to get total value of only 3 months of total order value, then we might have to use a subquery for the same.

```
Select ct.customer_id, sum(po.price_per_unit *
po.order_qty) as 'total_value’
from customers ct inner join orders od
on ct.customer_id = od.customer_id
inner join product_orders po
on od.order_no = po.order_no
and od.order_date >=
date_sub(current_date(),INTERVAL,3,MONTH) and
ct.customer_id in
(
Select ct.customer_id,
from customers ct left outer join orders od
on ct.customer_id = od.customer_id
left outer join product_orders po
on od.order_no = po.order_no
group by ct.customer_id, od.order_no
order by sum(po.price_per_unit * po.order_qty) as
'total_value'
Limit 10
 )
```

d. For all those orders which have been returned either for refund or exchange, identify the product id and get the list of other orders of those products, their customer reviews.

```
Solution:
select od.order_no, customer_review from
orders od, product_orders po
Where od.order_status not in ("REFUND", "EXCHANGE")
and od.order_no = po.order_no and
po.prod_id in
(select prod_id from
orders od, product_orders po
where od.order_no = po.order_no
and order_status = "REFUND" or order_status = "EXCHANGE")
```

7. Data Manipulation – Subqueries, SET Operators
   a. For all customers residing in metropolis cities as well as customers with age lesser than 25 years, find the order wise count of products as well as their total value
      Solution:
      1. First you need to capture city details separately for the customer. Also you need to define whether a city is metropolis in a separate table.
         Go ahead using Alter table add city column with varchar(20).
         Create another table name city_Metropolis which contains city varchar(20), Metropolis_flag boolean.

```
ALTER TABLE `customers`
ADD COLUMN `city` VARCHAR(20) NULL AFTER `cust_pincode`;

CREATE TABLE CITY_METROPOLIS(
CITY VARCHAR(20),
Metropolis_flag BOOLEAN);
```

      2. Now using the SQL joins and union operator, lets try to get the solution.

```
select   od.order_no,   count(distinct   po.prod_id),
sum(po.order_qty * po.price_per_unit)
from orders od, product_orders po
where od.order_no = po.order_no
and od.customer_id in
(
select ct.customer_id from
customers ct inner join city_metropolis cy
on ct.city = cy.city
and cy.metropolis_flag = TRUE
Union
Select ct.customer_id from
customers ct where
(year(current_date()) - year(cust_dateofBirth) ) <= 25
)
group by od.order_no, po.prod_id;
```

   b. We need to get the list of customers ordering more than 10000 INR per month as well as list of customers who are ordering less than 5000 INR but products more than 40 different types as well as customers who order more than 5000 lesser than 10000 and near 20 different types of products.

```
select od.customer_id
from product_orders po , orders od
```

```
where od.order_no = po.order_no
group by od.customer_id, month(od.order_date)
having sum(po.order_qty * po.price_per_unit) >= 10000
union
select od.customer_id
from product_orders po , orders od,
prod_type pt , products pd
where od.order_no = po.order_no
and pt.prod_type = pd.prod_type
and pd.prod_id = po.prod_id
group by od.customer_id, month(od.order_date)
having sum(po.order_qty * po.price_per_unit) < 5000
and count(distinct po.prod_id) > 40
Union
select od.customer_id
from product_orders po , orders od,
prod_type pt , products pd
where od.order_no = po.order_no
and pt.prod_type = pd.prod_type
and pd.prod_id = po.prod_id
group by od.customer_id, month(od.order_date)
having sum(po.order_qty * po.price_per_unit) BETWEEN 5000
and 10000
and count(distinct po.prod_id) > 20
```

c. We need to get list of products that are highly popular in a region by way of sales as well as list of products which are highest review ratings in the same region.

```
Solution Hint :
Here you write the two queries and use the intersect
operator.
```

## Appendix – 1: Sample Database Structure

Database Name : MyShop

```
-- Drop all the tables if it already exists:
drop table IF EXISTS prod_type;
drop table IF EXISTS product_orders;
drop table IF EXISTS orders;
drop table IF EXISTS products;
drop table IF EXISTS customers;

-- Table : customers
create table customers(
customer_id varchar(20) NOT NULL ,
cust_first_name varchar(20),
cust_middle_name varchar(20),
cust_last_name varchar(20),
cust_phone varchar(15),
cust_address_1 varchar(50),
cust_pincode varchar(10),
cust_state varchar(15),
cust_country varchar(20),
cust_dateOfBirth date,
primary key (customer_id)
);


-- Table : Products
create table products(
prod_id varchar(10) not null ,
prod_type varchar(10) not null,
prod_name varchar(30) not null,
prod_desc varchar(100),
prod_unit varchar(5),
prod_available_qty int,
prod_price_unit decimal(10,2) NOT NULL,
prod_status varchar(10) NOT NULL DEFAULT 'AVL',
Primary Key (prod_id)
);

-- Table : Orders
CREATE TABLE orders (
  order_no bigint NOT NULL AUTO_INCREMENT,
  customer_id varchar(20) NOT NULL,
  order_date date NOT NULL,
  customer_review tinyint,
  order_status varchar(15) NOT NULL DEFAULT 'PLACED',
  PRIMARY KEY (`order_no`),
  CONSTRAINT orders_ibfk_1 FOREIGN KEY (`customer_id`) REFERENCES
`customers` (`customer_id`)
) ENGINE=InnoDB;

-- Table product_orders
```

```
CREATE TABLE product_orders (
  order_no bigint NOT NULL,
  prod_id varchar(15) NOT NULL,
  order_qty int NOT NULL,
  price_per_unit decimal(10,2) NOT NULL,
 PRIMARY KEY (`order_no`,`prod_id`),
  CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`order_no`) REFERENCES
`orders` (`order_no`),
  CONSTRAINT `orders_ibfk_3` FOREIGN KEY (`prod_id`) REFERENCES
`products` (`prod_id`)
) ENGINE=InnoDB;

-- Table prod_type
CREATE TABLE `prod_type` (
  prod_type varchar(10) NOT NULL,
  prod_type_desc varchar(20) NOT NULL,
  prod_type_uom varchar(15) NOT NULL,
  PRIMARY KEY (`prod_type`)
) ENGINE=InnoDB;
```