islington college
(इस्लिङ्टन कलेज)

LONDON METROPOLITAN UNIVERSITY

Model Code & Module Titel

CC4002NA Information Systems

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2019 spring

Student Name: Bikram kumar Sharma

London Met ID:18030830

Collage ID: np01cp4s190050

Assignment Due Date: 2019/07/19

Assignment Submission Date: 2019/07/19

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Proposal

Introduction:

The given project is all about writing the Queries about database in which any company or organization can store their data or information. Here I create the queries about a company AndroidStore, Where I store different data like name of Customer, Name of Items, and Delivery.

Problem Statement:

While managing data of different people or items of Store, many problem might be faced by Suppliers, if there is no more use of Database. The most problem faced by them are when someone return their Items and said to return their money, when asked for see their Ordered date, etc. If there is no any data information, It is problem for suppliers to ask about Items to Delivery if they delivered or not.

Evaluating this scope (Aims and Objectives):

As this project consists of five different tables in which we have to write different queries. The main aims and objectives of this coursework is to complete by 19th July (week 21). This module includes a simple way to motivate the students in area of XAMMP's queries. This project consists of database research in which we have to write different queries on the basic of module. The main objective of the module was to make students more experiences in the field for overcoming the error while writing queries.

Proposed Approach:

As when we create a system for store data, then there we might face less problem. When any company create their own Database to store their information, their company will growth. Here when I create database to store my information in many related tables, I face quite easy to get information of Items, Suppliers, Delivery, and Customers because whenever I need data I easily find.

Target Audience:

The project that was given as a coursework might be very helpful for everyone, maybe for all small to big organizations which are running whether for commercial or non-commercial purpose because of the flexibility of storing and it manages data efficiently and allows users to perform multiple tasks with ease. This project is also helpful for students for designing database queries, populate fee database and implement a database. Overall, this is very helpful.

Hardware and Software Requirements:

The software that were required for this project are XAMMP, the community version of MySQL.

Activity Description and Timeline:

Time duration	Activities involved
Week 17	Coursework was handed. Discussion about coursework was done with the help of module teacher after that research was Started.
Week 18	Writing queries was done by the students by the help of Module teacher.
Week 19	Difficulties while writing queries were ask by the students to the module teacher checked by module teacher.
Week 20	Documentation, Data dictionary, Screenshot of different queries and referencing was carried out the project.
Week 21	Research were done on different sides, Books and finally project was submitted

Table of Contents:

Table of tables:

Tables of Figures:

1. Introduction:

Data is the collection of facts and figures, such as number, words, measurements, observations or even just descriptions of things. Database is the collection of information that is organized so that it can be easily accessed, managed and updated. It is collection of tables and modern databases use fourth-generation language (4GL) which allows user to specify what must be done without specifying how it is to be done. Relational database management system is a common type of database whose data is stored in tables. We'll find most database used in businesses these days are relational databases, as opposed to a flat file hierarchical database (Unknown2, n.d.).

As this coursework is all about the creating the database of any organization or company. Here I create the database about any mobile shop, where there are many Suppliers which have different Items of Mobiles. At first Suppliers have Items of mobile in their shop, where customers used to give their order and salesman sales their items and finally Items delivered by Delivery. This project generally give information about suppliers, Items, Orders, Customer and Delivery that how much items has sales, which Suppliers supplies their Items, Which Items is delivered by which Delivery and which Customers gives their order.

In the given project, there are five different tables (relations) Suppliers, Items, Order, Customers, and Delivery. In each tables there are suitable primary key described with suitable attributes (columns) and each attributes are constrained by using suitable constraints (i.e. unique, not null, auto increments, etc.). Each relations is interlinked by using suitable pairing of foreign keys and tables is illustrated by using suitable diagram (i.e. entity-relationship diagram and relation diagram). There are different screenshot which display the different tables and data present in the database. There are different Data Dictionary which describes the structure of the whole database and provide metadata, or information about data. This project includes ten different queries with appropriate screenshot. This project is very useful to store customers and salesman information, it manages data efficiently and allows users to perform multiple tasks

with ease. It provide a highly efficient method for handling multiple types of data, by this project both salesman and customers easy.

2. Discussion and analysis:

In the given coursework, the project is all about writing queries on different order items in shop. This coursework had given many tasks such as creating suitable primary, foreign, unique keys, creating data dictionary, which help salesman and customers to find their details on items. Any how this project is somehow difficult to finish but with hard effort, project is successful.

Relational database are collection of one or more relations, in practice relations are tables, the rows of which are individual records of data with same structure. It uses a structure that allows us to identify and access data in relation to another pieces of data in the database. Every table shares at least one field with another table in 'one to one', 'one to many', or 'many to many' relationships (unknown3, n.d.).

Database Management System (DBMS) is a system software for creating and managing databases. It provides users and programmers with a systematic way to create, retrieve, update and manage data and acts as facilitator/interface for us to interact database. It controls data redundancy, data Independence, control security (unknown3, n.d.).

Data Integrity validates the data before getting stored in the columns of the table. It is the overall completeness, accuracy and consistency of data. Entity integrity, Referential integrity and Domain integrity are the three constraints used in the database structure.

Primary key must contain a unique value for each row of data. It cannot contain null values, must be unique, not empty. In each table there can only one primary key.

Foreign key refers to the primary key in another table. It is used in order to relate two tables.

Unique key ensures that all values in a columns are different. Unique constraints can have null values and there can be more than one unique key in a table.

Auto Increment allows a unique number to be generated automatically when a new record is inserted into a table. The starting value of AUTO_INCREMENT is 1, and it will increment by 1 for each new record. It generating a unique primary key in each table.

Structured Query Language (SQL) is a standard language for dealing with relation database. It is used to insert, search, update and delete database records. MySQL is its syntax. MySQL is a very popular, open source database which handles very large database with fast performance. It is easy to use shell for creating tables, querying tables, etc. XAMMP is a free and open source cross-platform web server solution package developed by Apache. It stands for Cross-platform Apache MariaDB PHP Perl (XAMMP) and is a community version of MySQL. It extremely easy for developers to create a local web server for testing and deployment purposes. MySQL is run by using XAMMP (unknown1, n.d.).

I create my Database in the name of AndroidStore, in which there are five table or relations. At first I create Suppliers table because it supplies items. Then Items table is created because Items have Orders and in Items table their exists primary key of Suppliers as foreign key name as SuppliersID. After that Orders table is created because Orders makes Customer in which their exist two foreign key, then customer table is created and Finally Orders delivered by delivery and Delivery table is created.

```
MariaDB [(none)]> create database AndroidStore;
Query OK, 1 row affected (0.009 sec)
```

3. Entity-RelationShip Diagram:

An entity-relationship diagram (ERD) is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within that system. ERDs are used for data modeling which can help define business processes and can be used as the foundation for a relational database (unknown4, n.d.).

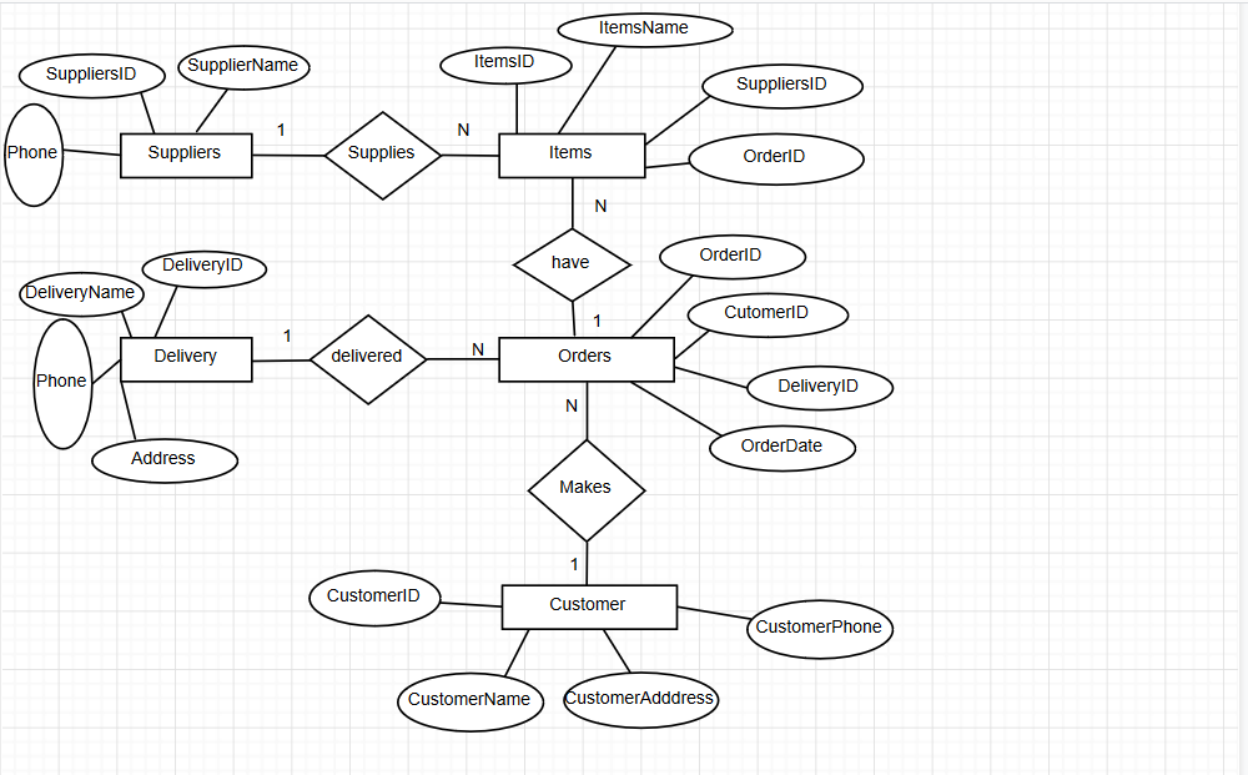

Fig1: ERD of AndroidStore database

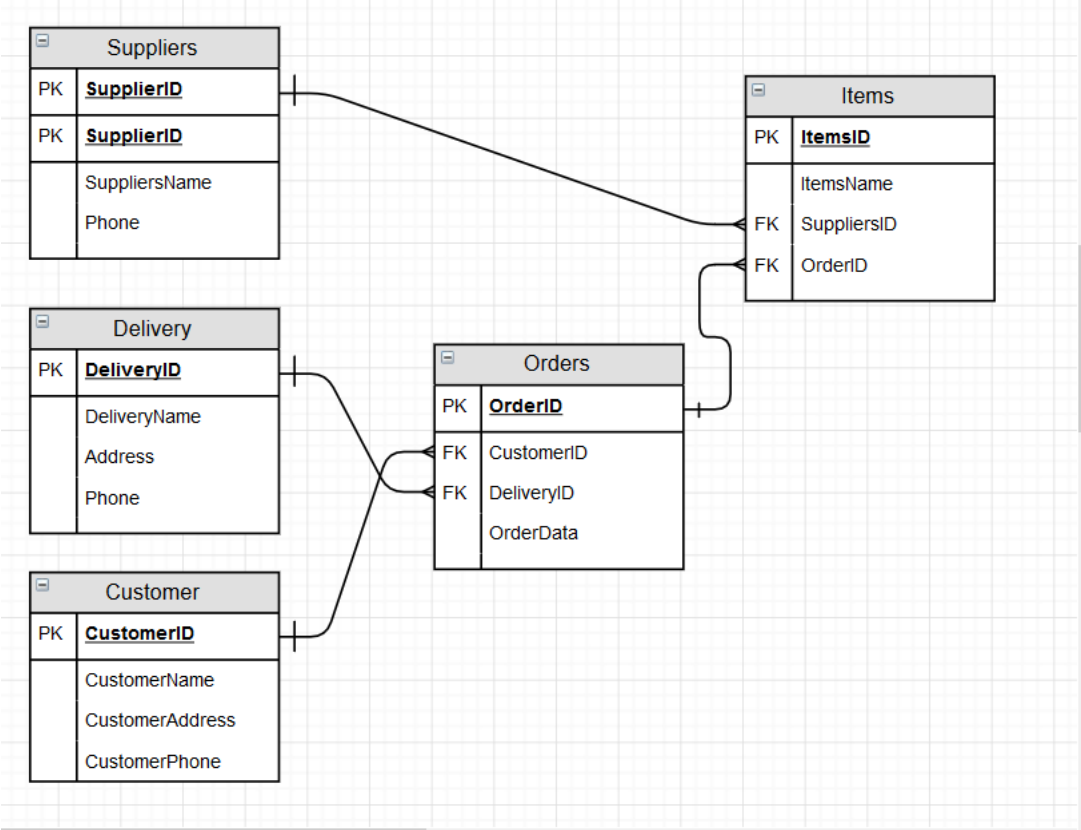

Fig1: Relation diagram of AndroidStore database

4. DataBase model:

a) For Suppliers:

At first Suppliers table is created in which there are three attributes (SupplierID, SuppliersName, and Phone), among them SuppliersID is taken as primary key because this primary key reference the attributes of Items tables , supplierName is Suppliers who supplies Items and phone is the phone Number of Suppliers.

```
MariaDB [AndroidStore]> Create table Suppliers ( SuppliersID int Primary key, SuppliersName varchar(255), Phone varchar(
255));
Query OK, 0 rows affected (0.054 sec)
```

```
MariaDB [AndroidStore]> desc Suppliers;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| SuppliersID   | int(11)     | NO   | PRI | NULL    |       |
| SuppliersName | varchar(255) | YES  |     | NULL    |       |
| Phone         | varchar(255) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
3 rows in set (0.043 sec)
```

```
MariaDB [AndroidStore]>  insert into Suppliers values ( 1, "Samsung", "01-4104677"), (2, "Techno", "055-422256"), (3, "N
okia", "044-526576"), (4, "Oppo", "055-675439"), (5, "Vivo", "041-765328");
Query OK, 5 rows affected (0.010 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
MariaDB [AndroidStore]> select*from Suppliers;
+-------------+---------------+------------+
| SuppliersID | SuppliersName | Phone      |
+-------------+---------------+------------+
|           1 | Samsung       | 01-4104677 |
|           2 | Techno        | 055-422256 |
|           3 | Nokia         | 044-526576 |
|           4 | Oppo          | 055-675439 |
|           5 | Vivo          | 041-765328 |
+-------------+---------------+------------+
5 rows in set (0.000 sec)
```

b) For Items:

As Suppliers supplies Items, so Items table is created after Suppliers table. So, in items table there are four columns (ItemsID, ItemsName, SuppliersID, OrderID), among them ItemsID Is taken as primary key because this attributes give items number and SuppliersID and OrderID are taken as FOREIGN KEY because SupplierID is REFERENCES by the column of Suppliers and OrderID is REFERENCES by the column of OrderID respectively. And ItemsAName is the name of Items present in Store.

```
MariaDB [AndroidStore]> create table Items ( ItemsID int primary key, ItemsName varchar(255), SuppliersID int, FOREIGN K
EY (SuppliersID) REFERENCES Suppliers (SuppliersID), OrderId int, FOREIGN KEY (OrderID) REFERENCES Orders (OrderId));
Query OK, 0 rows affected (0.042 sec)
```

```
MariaDB [AndroidStore]> describe Items;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| ItemsID     | int(11)      | NO   | PRI | NULL    |       |
| ItemsName   | varchar(255) | YES  |     | NULL    |       |
| SuppliersID | int(11)      | YES  | MUL | NULL    |       |
| OrderId     | int(11)      | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.025 sec)
```

```
MariaDB [AndroidStore]> insert into Items values (1, "Samsung S7", 1, 2), (2, "Nokia 7 Plus", 3, 6), (3, "Samsung Galaxy
", 1, 5), (4, "Techno camon I", 2, 3), (5, "Oppo A+", 4, 1), (6, "Samsung S7", 1, 9), (7, "Techno Camon I", 2, 6), (8, "
Vivo V7", 5, 3);
Query OK, 8 rows affected (0.010 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

```
MariaDB [AndroidStore]> select*from Items;
+---------+---------------+-------------+---------+
| ItemsID | ItemsName     | SuppliersID | OrderId |
+---------+---------------+-------------+---------+
|       1 | Samsung S7    |           1 |       2 |
|       2 | Nokia 7 Plus  |           3 |       6 |
|       3 | Samsung Galaxy|           1 |       5 |
|       4 | Techno camon I|           2 |       3 |
|       5 | Oppo A+       |           4 |       1 |
|       6 | Samsung S7    |           1 |       9 |
|       7 | Techno Camon I|           2 |       6 |
|       8 | Vivo V7       |           5 |       3 |
+---------+---------------+-------------+---------+
8 rows in set (0.000 sec)
```

c) **For Orders:**

After Items table, Orders table is created because Items have Orders. So in Orders table there are four columns (OrderID, CustomerID, DeliveryID, OrderDate), amoing them OrderID attributes is taken as primary key because this attributes give REFERENCES to table Items. CustomerID and DeliveryID are taken as FOREIGN KEY because CustomerID is REFERENCES by the column of Customer and DeliveryID is REFERENCES by the column of Delivery respectivrly and Orderdate is the date when customer makes order.

```
MariaDB [AndroidStore]> create table Orders (OrderID int Primary key, CustomerID int, FOREIGN KEY (CustomerID) REFERENCE
S Customer (CustomerID), DeliveryID int, FOREIGN KEY (DeliveryID) REFERENCES Delivery (DeliveryID), OrderDate varchar(25
5));
Query OK, 0 rows affected (0.044 sec)
```

```
MariaDB [AndroidStore]> desc Orders;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| OrderID    | int(11)      | NO   | PRI | NULL    |       |
| CustomerID | int(11)      | YES  | MUL | NULL    |       |
| DeliveryID | int(11)      | YES  | MUL | NULL    |       |
| OrderDate  | varchar(255) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.020 sec)
```

```
MariaDB [AndroidStore]> insert into Orders values (1, 1, 6, "2018-01-10"), (2, 4, 2, "2018-01-25"), (3, 6, 4, "2018-02-2
1"), (4, 3, 3, "2018-03-03"), (5, 2, 1, "2018-03-26"), (6, 5, 6, "2018-04-09"), (7, 1, 5, "2018-05-21"), (8, 2, 3, "2018
-06-11"), (9, 3, 6, "2018-07-15"), (10, 6, 4, "2018-08-09");
Query OK, 10 rows affected (0.011 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
MariaDB [AndroidStore]> select*from Orders;
+---------+------------+------------+------------+
| OrderID | CustomerID | DeliveryID | OrderDate  |
+---------+------------+------------+------------+
|       1 |          1 |          6 | 2018-01-10 |
|       2 |          4 |          2 | 2018-01-25 |
|       3 |          6 |          4 | 2018-02-21 |
|       4 |          3 |          3 | 2018-03-03 |
|       5 |          2 |          1 | 2018-03-26 |
|       6 |          5 |          6 | 2018-04-09 |
|       7 |          1 |          5 | 2018-05-21 |
|       8 |          2 |          3 | 2018-06-11 |
|       9 |          3 |          6 | 2018-07-15 |
|      10 |          6 |          4 | 2018-08-09 |
+---------+------------+------------+------------+
10 rows in set (0.000 sec)
```

d) For Customer:

As Orders makes Customer, so Customer table is created after Order table. In Customer table there are four attributes (CustomerID, CustomerName, Customeraddress, CustomerPhone) among them CustomerID Is taken as primary key because it gives REFERENCES to Order table and CustomerName is Name of Customer, CustomerAddress is the address of Customer and CustomerPhone is the Phone of Customer.

```
MariaDB [AndroidStore]> create table Customer (CustomerID int primary key, CustomerName varchar(255), CustomerAddress va
rchar(255), CustomerPhone varchar(255));
Query OK, 0 rows affected (0.039 sec)
```

```
MariaDB [AndroidStore]> describe Customer;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| CustomerID      | int(11)      | NO   | PRI | NULL    |       |
| CustomerName    | varchar(255) | YES  |     | NULL    |       |
| CustomerAddress | varchar(255) | YES  |     | NULL    |       |
| CustomerPhone   | varchar(255) | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
4 rows in set (0.026 sec)
```

```
MariaDB [AndroidStore]> Insert into Customer values (1, "Sunny Yadav", "Bara", "9801004243"), (2, "Randhir Chaurasiya",
"Rauthat", "9811104233"), (3, "Mukesh Shah", "Rauthat", "9844004243"), (4, "Durgesh Singh", "Sarlahi", "9842430243"), (5
, "Parash Mani", "Siraha", "9814872146"), (6, "Avhay Singh", "Mohattari", "9812002326");
Query OK, 6 rows affected (0.010 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
MariaDB [AndroidStore]> select*from Customer;
+------------+--------------------+-----------------+---------------+
| CustomerID | CustomerName       | CustomerAddress | CustomerPhone |
+------------+--------------------+-----------------+---------------+
|          1 | Sunny Yadav        | Bara            | 9801004243    |
|          2 | Randhir Chaurasiya | Rauthat         | 9811104233    |
|          3 | Mukesh Shah        | Rauthat         | 9844004243    |
|          4 | Durgesh Singh      | Sarlahi         | 9842430243    |
|          5 | Parash Mani        | Siraha          | 9814872146    |
|          6 | Avhay Singh        | Mohattari       | 9812002326    |
+------------+--------------------+-----------------+---------------+
6 rows in set (0.000 sec)
```

e) For Delivery:

As Order have Delivery, so Delivery table is created after Order table to connect each other. In Delivery table there are Four columns (DeliveryID, DeliveryName, Address, phone), among them Delivery column is taken as primary key because it gives REFERENCES to table Orders. And DeliveryName is the name of Delivery, address is the name of delivery address and phone is the phone number of delivery.

```
MariaDB [AndroidStore]> Create table Delivery ( DeliveryID int primary key, DeliveryName varchar(255), Address varchar(2
55), Phone varchar(255));
Query OK, 0 rows affected (0.037 sec)
```

```
MariaDB [AndroidStore]> desc Delivery;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| DeliveryID   | int(11)      | NO   | PRI | NULL    |       |
| DeliveryName | varchar(255) | YES  |     | NULL    |       |
| Address      | varchar(255) | YES  |     | NULL    |       |
| Phone        | varchar(255) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
4 rows in set (0.027 sec)
```

```
MariaDB [AndroidStore]> Insert into Delivery values ( 1, "Bikesh Yadav", "Rauthat", "9854011222"), (2, "Amresh Shah", "J
hapa", "9865021499"), (3, "Razan pal", "Nuwakot", "9855521912"), (4, "Aakash Chaudhari", "Morang", "9865491711"), (5, "H
emanth Thakur", "Kathmandu", "9854032434"), (6, "Sidhratha poudel", "Nuwakot", "9842465780");
Query OK, 6 rows affected (0.011 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
MariaDB [AndroidStore]> select*from Delivery;
+------------+-----------------+-----------+------------+
| DeliveryID | DeliveryName    | Address   | Phone      |
+------------+-----------------+-----------+------------+
|          1 | Bikesh Yadav    | Rauthat   | 9854011222 |
|          2 | Amresh Shah     | Jhapa     | 9865021499 |
|          3 | Razan pal       | Nuwakot   | 9855521912 |
|          4 | Aakash Chaudhari | Morang   | 9865491711 |
|          5 | Hemanth Thakur  | Kathmandu | 9854032434 |
|          6 | Sidhratha poudel | Nuwakot  | 9842465780 |
+------------+-----------------+-----------+------------+
6 rows in set (0.000 sec)
```

5. Data Dictionary:

Proper formatted of data dictionary for each table of the database are given below:

a. For Suppliers:

Entity Name	Entity Description	Column Name	Column Description	Data types	Length	Primary key	Foreign key	Nullable	Unique	Notes
Suppliers	Suppliers are someone who supplies items	SuppliersID	ID of the Suppliers for unique identification of Suppliers	Int	11	True	False	False	True	
		Suppliers Name	Name of the suppliers	Varchar	255	False	True	False	False	
		Phone	Phone of the Suppliers	varchar	255	False	True	False	False	

Table1 : data dictionary of Suppliers table

b. For Items:

Entity Name	Entity Description	Column Name	Column Description	Data types	Length	Primary key	Foreign key	Nullable	Unique	Notes
Items	Items is a things which makes orders	ItemsID	Id of Items for unique identification	Int	11	True	False	False	True	
		ItemsName	Name of the Items	varchar	255	false	False	False	False	
		SuppliersID	Id of suppliers who supplies Items	int	11	False	True	False	False	
		OrderID	Id of orders, who gives order on items	int	11	False	True	False	False	

Table2 : data dictionary of Items table

c. For Orders:

Entity Name	Entity Description	Column Name	Column Description	Data Types	Length	Primary key	Foreign key	Nullable	Unique	notes
Orders	It is the process Which have Items	OrderID	ID of order gives unique identification of order given by Customer	Int	11	True	False	False	True	
		CustomerID	ID of Customer gives unique identification of Customer	Int	11	False	True	False	False	
		DeliveryID	ID of Delivery gives unique identification of Delivery	Int	11	False	True	False	False	
		OrderDate	Date of Order given by Customer	varchar	255	False	False	False	False	

Table3 : data dictionary of Orders table

d. For Customer:

Entity Name	Entity Description	Column Name	Column Description	Data types	Length	Primary key	Foreign key	Nullable	Unique	Notes
Customer	Customer is someone how gives orders on items	CustomerID	ID of Customer for unique identification	Int	11	True	False	False	True	
		CustomerName	Name of Customer	Varchar	255	False	False	False	False	
		CustomerAddress	Address of Customer	Varchar	255	False	False	False	False	
		CustomerPhone	Phone number of Customer	Varchar	255	False	False	False	False	

Table4 : data dictionary of Customer table

e. For Delivery:

Entity Name	Entity Description	Column Name	Column Description	Data types	Length	Primary key	Foreign key	Nullable	Unique	Notes
Delivery	Delivery are the person who delivered items to Customer	DeliveryID	Id of Delivery for unique identification	Int	11	True	False	False	True	
		DeliveryName	Name of Delivery	Varchar(255)	255	False	False	False	False	
		Address	Address of Delivery	Varchar(255)	255	False	False	False	False	
		Phone	Phone Number of delivery	Varchar(255)	255	False	False	False	False	

Table5 : data dictionary of Delivery table

6. Queries:

a. BETWEEN:

The BETWEEN operator selects values within a range. The values can be numbers, texts or dates.

```
MariaDB [AndroidStore]> select*from Delivery;
+------------+------------------+-----------+------------+
| DeliveryID | DeliveryName     | Address   | Phone      |
+------------+------------------+-----------+------------+
|          1 | Bikesh Yadav     | Rauthat   | 9854011222 |
|          2 | Amresh Shah      | Jhapa     | 9865021499 |
|          3 | Razan pal        | Nuwakot   | 9855521912 |
|          4 | Aakash Chaudhari | Morang    | 9865491711 |
|          5 | Hemanth Thakur   | Kathmandu | 9854032434 |
|          6 | Sidhratha poudel | Nuwakot   | 9842465780 |
+------------+------------------+-----------+------------+
6 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select*from delivery where deliveryID BETWEEN "1" and "5";
+------------+------------------+-----------+------------+
| DeliveryID | DeliveryName     | Address   | Phone      |
+------------+------------------+-----------+------------+
|          1 | Bikesh Yadav     | Rauthat   | 9854011222 |
|          2 | Amresh Shah      | Jhapa     | 9865021499 |
|          3 | Razan pal        | Nuwakot   | 9855521912 |
|          4 | Aakash Chaudhari | Morang    | 9865491711 |
|          5 | Hemanth Thakur   | Kathmandu | 9854032434 |
+------------+------------------+-----------+------------+
5 rows in set (0.010 sec)
```

b. IN:

The IN operator allows to specify multiple values in a WHERE clause. It performs membership check.

```
MariaDB [AndroidStore]> select*from Suppliers;
+-------------+--------------+------------+
| SuppliersID | SuppliersName | Phone      |
+-------------+--------------+------------+
|           1 | Samsung       | 01-4104677 |
|           2 | Techno        | 055-422256 |
|           3 | Nokia         | 044-526576 |
|           4 | Oppo          | 055-675439 |
|           5 | Vivo          | 041-765328 |
+-------------+--------------+------------+
5 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select*from Suppliers where SuppliersID IN (1, 2, 3, 4);
+-------------+--------------+------------+
| SuppliersID | SuppliersName | Phone      |
+-------------+--------------+------------+
|           1 | Samsung       | 01-4104677 |
|           2 | Techno        | 055-422256 |
|           3 | Nokia         | 044-526576 |
|           4 | Oppo          | 055-675439 |
+-------------+--------------+------------+
4 rows in set (0.008 sec)
```

c. LIKE:

The LIKE operator is used in a WHERE clause to search for a specified pattern.

```
MariaDB [AndroidStore]> select*from Items;
+---------+----------------+-------------+---------+
| ItemsID | ItemsName      | SuppliersID | OrderId |
+---------+----------------+-------------+---------+
|       1 | Samsung S7     |           1 |       2 |
|       2 | Nokia 7 Plus   |           3 |       6 |
|       3 | Samsung Galaxy |           1 |       5 |
|       4 | Techno camon I |           2 |       3 |
|       5 | Oppo A+        |           4 |       1 |
|       6 | Samsung S7     |           1 |       9 |
|       7 | Techno Camon I |           2 |       6 |
|       8 | Vivo V7        |           5 |       3 |
+---------+----------------+-------------+---------+
8 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select*from Items where ItemsName like "%sa%";
+---------+----------------+-------------+---------+
| ItemsID | ItemsName      | SuppliersID | OrderId |
+---------+----------------+-------------+---------+
|       1 | Samsung S7     |           1 |       2 |
|       3 | Samsung Galaxy |           1 |       5 |
|       6 | Samsung S7     |           1 |       9 |
+---------+----------------+-------------+---------+
3 rows in set (0.000 sec)
```

d. ORDER BY:

ORDER BY sorts the records in ascending or descending order according to the specified column.

```
MariaDB [AndroidStore]> select*from Delivery;
+------------+------------------+-----------+------------+
| DeliveryID | DeliveryName     | Address   | Phone      |
+------------+------------------+-----------+------------+
|          1 | Bikesh Yadav     | Rauthat   | 9854011222 |
|          2 | Amresh Shah      | Jhapa     | 9865021499 |
|          3 | Razan pal        | Nuwakot   | 9855521912 |
|          4 | Aakash Chaudhari | Morang    | 9865491711 |
|          5 | Hemanth Thakur   | Kathmandu | 9854032434 |
|          6 | Sidhratha poudel | Nuwakot   | 9842465780 |
+------------+------------------+-----------+------------+
6 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select*from Delivery ORDER BY DeliveryName;
+------------+------------------+-----------+------------+
| DeliveryID | DeliveryName     | Address   | Phone      |
+------------+------------------+-----------+------------+
|          4 | Aakash Chaudhari | Morang    | 9865491711 |
|          2 | Amresh Shah      | Jhapa     | 9865021499 |
|          1 | Bikesh Yadav     | Rauthat   | 9854011222 |
|          5 | Hemanth Thakur   | Kathmandu | 9854032434 |
|          3 | Razan pal        | Nuwakot   | 9855521912 |
|          6 | Sidhratha poudel | Nuwakot   | 9842465780 |
+------------+------------------+-----------+------------+
6 rows in set (0.009 sec)
```

e. LIMIT:

LIMIT is used to specify the number of records to show as the result of a query.

```
MariaDB [AndroidStore]> select*from Delivery;
+------------+------------------+-----------+------------+
| DeliveryID | DeliveryName     | Address   | Phone      |
+------------+------------------+-----------+------------+
|          1 | Bikesh Yadav     | Rauthat   | 9854011222 |
|          2 | Amresh Shah      | Jhapa     | 9865021499 |
|          3 | Razan pal        | Nuwakot   | 9855521912 |
|          4 | Aakash Chaudhari | Morang    | 9865491711 |
|          5 | Hemanth Thakur   | Kathmandu | 9854032434 |
|          6 | Sidhratha poudel | Nuwakot   | 9842465780 |
+------------+------------------+-----------+------------+
6 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select*from Delivery ORDER BY DeliveryName Limit 3;
+------------+------------------+---------+------------+
| DeliveryID | DeliveryName     | Address | Phone      |
+------------+------------------+---------+------------+
|          4 | Aakash Chaudhari | Morang  | 9865491711 |
|          2 | Amresh Shah      | Jhapa   | 9865021499 |
|          1 | Bikesh Yadav     | Rauthat | 9854011222 |
+------------+------------------+---------+------------+
3 rows in set (0.000 sec)
```

f. DISTINCT:

DISTINCT is an aggregate function used to return only the unique/distinct entries from a column.

```
MariaDB [AndroidStore]> select*from Customer;
+------------+--------------------+-----------------+---------------+
| CustomerID | CustomerName       | CustomerAddress | CustomerPhone |
+------------+--------------------+-----------------+---------------+
|          1 | Sunny Yadav        | Bara            | 9801004243    |
|          2 | Randhir Chaurasiya | Rauthat         | 9811104233    |
|          3 | Mukesh Shah        | Rauthat         | 9844004243    |
|          4 | Durgesh Singh      | Sarlahi         | 9842430243    |
|          5 | Parash Mani        | Siraha          | 9814872146    |
|          6 | Avhay Singh        | Mohattari       | 9812002326    |
+------------+--------------------+-----------------+---------------+
6 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select Distinct (CustomerAddress) from Customer;
+-----------------+
| CustomerAddress |
+-----------------+
| Bara            |
| Rauthat         |
| Sarlahi         |
| Siraha          |
| Mohattari       |
+-----------------+
5 rows in set (0.000 sec)
```

g. COUNT:

```
MariaDB [AndroidStore]> select*from Orders;
+---------+------------+------------+------------+
| OrderID | CustomerID | DeliveryID | OrderDate  |
+---------+------------+------------+------------+
|       1 |          1 |          6 | 2018-01-10 |
|       2 |          4 |          2 | 2018-01-25 |
|       3 |          6 |          4 | 2018-02-21 |
|       4 |          3 |          3 | 2018-03-03 |
|       5 |          2 |          1 | 2018-03-26 |
|       6 |          5 |          6 | 2018-04-09 |
|       7 |          1 |          5 | 2018-05-21 |
|       8 |          2 |          3 | 2018-06-11 |
|       9 |          3 |          6 | 2018-07-15 |
|      10 |          6 |          4 | 2018-08-09 |
+---------+------------+------------+------------+
10 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select count(*) as OrderID from orders;
+---------+
| OrderID |
+---------+
|      10 |
+---------+
1 row in set (0.008 sec)
```

COUNT(*) returns the total number of row in a table. COUNT(column) returns the total number of records in the column excluding NULL values.

h. GROUP BY:

GROUP BY is used to group the records according to similar values in a certain column.

```
MariaDB [AndroidStore]> select*from Items;
+---------+----------------+-------------+---------+
| ItemsID | ItemsName      | SuppliersID | OrderId |
+---------+----------------+-------------+---------+
|       1 | Samsung S7     |           1 |       2 |
|       2 | Nokia 7 Plus   |           3 |       6 |
|       3 | Samsung Galaxy |           1 |       5 |
|       4 | Techno camon I |           2 |       3 |
|       5 | Oppo A+        |           4 |       1 |
|       6 | Samsung S7     |           1 |       9 |
|       7 | Techno Camon I |           2 |       6 |
|       8 | Vivo V7        |           5 |       3 |
+---------+----------------+-------------+---------+
8 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select SuppliersID, count(*) as TotalSupplied from Items GROUP BY SuppliersID;
+-------------+--------------+
| SuppliersID | TotalSupplied |
+-------------+--------------+
|           1 |            3 |
|           2 |            2 |
|           3 |            1 |
|           4 |            1 |
|           5 |            1 |
+-------------+--------------+
5 rows in set (0.000 sec)
```

i. HAVING:

Similar functionality as WHERE, but HAVING is used to filter data returned by aggregated function as Where cannot do so.

```
MariaDB [AndroidStore]> select*from Items;
+---------+----------------+-------------+---------+
| ItemsID | ItemsName      | SuppliersID | OrderId |
+---------+----------------+-------------+---------+
|       1 | Samsung S7     |           1 |       2 |
|       2 | Nokia 7 Plus   |           3 |       6 |
|       3 | Samsung Galaxy |           1 |       5 |
|       4 | Techno camon I |           2 |       3 |
|       5 | Oppo A+        |           4 |       1 |
|       6 | Samsung S7     |           1 |       9 |
|       7 | Techno Camon I |           2 |       6 |
|       8 | Vivo V7        |           5 |       3 |
+---------+----------------+-------------+---------+
8 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select SuppliersID, count(*) as TotalSupplied from Items GROUP BY SuppliersID HAVING TotalSuppli
ed <3;
+-------------+--------------+
| SuppliersID | TotalSupplied |
+-------------+--------------+
|           2 |            2 |
|           3 |            1 |
|           4 |            1 |
|           5 |            1 |
+-------------+--------------+
4 rows in set (0.000 sec)
```

j. JOIN:

Used to join two or more tables based on similar columns.

```
MariaDB [AndroidStore]> select*from Orders;
+---------+------------+------------+------------+
| OrderID | CustomerID | DeliveryID | OrderDate  |
+---------+------------+------------+------------+
|       1 |          1 |          6 | 2018-01-10 |
|       2 |          4 |          2 | 2018-01-25 |
|       3 |          6 |          4 | 2018-02-21 |
|       4 |          3 |          3 | 2018-03-03 |
|       5 |          2 |          1 | 2018-03-26 |
|       6 |          5 |          6 | 2018-04-09 |
|       7 |          1 |          5 | 2018-05-21 |
|       8 |          2 |          3 | 2018-06-11 |
|       9 |          3 |          6 | 2018-07-15 |
|      10 |          6 |          4 | 2018-08-09 |
+---------+------------+------------+------------+
10 rows in set (0.000 sec)
```

```
MariaDB [AndroidStore]> select*from Customer;
+------------+--------------------+-----------------+---------------+
| CustomerID | CustomerName       | CustomerAddress | CustomerPhone |
+------------+--------------------+-----------------+---------------+
|          1 | Sunny Yadav        | Bara            | 9801004243    |
|          2 | Randhir Chaurasiya | Rauthat         | 9811104233    |
|          3 | Mukesh Shah        | Rauthat         | 9844004243    |
|          4 | Durgesh Singh      | Sarlahi         | 9842430243    |
|          5 | Parash Mani        | Siraha          | 9814872146    |
|          6 | Avhay Singh        | Mohattari       | 9812002326    |
+------------+--------------------+-----------------+---------------+
6 rows in set (0.000 sec)
```

```
MariaDB [androidstore]> select * from Customer JOIN orders on customer.CustomerID = orders.customerID;
+------------+--------------------+-----------------+---------------+---------+------------+------------+------------+
| CustomerID | CustomerName       | CustomerAddress | CustomerPhone | OrderID | CustomerID | DeliveryID | OrderDate  |
+------------+--------------------+-----------------+---------------+---------+------------+------------+------------+
|          1 | Sunny Yadav        | Bara            | 9801004243    |       1 |          1 |          6 | 2018-01-10 |
|          4 | Durgesh Singh      | Sarlahi         | 9842430243    |       2 |          4 |          2 | 2018-01-25 |
|          6 | Avhay Singh        | Mohattari       | 9812002326    |       3 |          6 |          4 | 2018-02-21 |
|          3 | Mukesh Shah        | Rauthat         | 9844004243    |       4 |          3 |          3 | 2018-03-03 |
|          2 | Randhir Chaurasiya | Rauthat         | 9811104233    |       5 |          2 |          1 | 2018-03-26 |
|          5 | Parash Mani        | Siraha          | 9814872146    |       6 |          5 |          6 | 2018-04-09 |
|          1 | Sunny Yadav        | Bara            | 9801004243    |       7 |          1 |          5 | 2018-05-21 |
|          2 | Randhir Chaurasiya | Rauthat         | 9811104233    |       8 |          2 |          3 | 2018-06-11 |
|          3 | Mukesh Shah        | Rauthat         | 9844004243    |       9 |          3 |          6 | 2018-07-15 |
|          6 | Avhay Singh        | Mohattari       | 9812002326    |      10 |          6 |          4 | 2018-08-09 |
+------------+--------------------+-----------------+---------------+---------+------------+------------+------------+
10 rows in set (0.013 sec)
```

7. Research:

The given coursework was not simple to complete it easily. The continuous of effort and research on the topics given in tasks helped to complete all the problems successfully. When on research different knowledge on different topics on topics for

coursework helped a lot for better project. Research helps to accomplish the given project in time.

The websites, journals and books which were used for completing this project are given below:

a. Web Sites:

1.

https://www.w3schools.com/.../sql_orderby....

Referencing this website, about ORDER BY function in SQL, I understand that how ORDER BY keyword works. It set the result in either ascending or either in descending order. It sorts the result in ascending order by default and in descending order, we use DESC keyword.

2.

www.mysqltutorial.org/mysql-distinct.aspx

Referencing this website, about DISTINCT function in SQL, I understood that DISTINCT is used to avoid duplicate row in SELECT statement. It eliminated the repetition data of column and give distinct data column.

3.

https://www.w3resource.com/.../count-funct...

Referencing this website, about COUNT function in SQL, I understood that COUNT function return the number of rows presents in table in the criteria of WHERE clause. If there are null column values then it return o.

4.

https://www.geeksforgeeks.org/sql-group-by/

Referencing this website, about GROUP BY function in SQL, I understood that GROUP BY function is used to arrange identical data into group with the help of some function. I.e. Particular column of same values is arranged into the identical group.

5.

www.postgresqltutorial.com/postgresql-hav...

Referencing this website, about HAVING function in SQL, I understood that HAVING is used after GROUP BY. At first GROUP BY is applied which arrange data into the identical group and then HAVING is used to filter the data as required.

b. Books:

1. **Database System Concepts:**
   This book is useful for beginner. This book give focus on relational algebra, design and query optimization. It give knowledge about how to write complex queries and define schemas. This book is better than any relational database text that I've come across.

2. **Theory of Relational DataBase:**
   This book is an excellent reference for researchers in the field. The things covered this book are relational algebra, functional dependencies, null values. The purpose of this book is to give students enough background in relational database theory to understand the current research being done in the field and helpful for graduate.

3. **Databases, Types, and the Relational Model:**
   This book is essential reading for database professionals, but it is helpful for the future direction of data and database management systems. It provides a precise, formal definition of an abstract model of data, to be considered as a foundation for the design of a DBMS and a database language.

4. **Database Fundamentals:**

   By this book we learn the basics of relational database theory and other information models. This book discusses database logical and physical design, and introduces us to the SQL language. This book is important for understanding different information models, getting the basic of the SQL language and for practice using hands on exercises.

5. **An introduction to Relational Database Theory:**

   This book delivers a thorough discussion of the foundations of the relational model of database design along with a systematic treatment of the formal theory for the model. This book introduces us to the theory of relational databases, and SQL, focusing on the application of that theory to the design of computer languages that properly embrace it. The book is intended for those studying relational databases as a part of a degree course in Information Technology (IT).

8. Conclusion:

The researches and finding the above concludes that doing queries in XAMMP is not so difficult but also not very easy. Here we create a database for a company/shop to store their data systematic. It contain five relation to store data separately and become very easy to search, give information of their items, customer, Suppliers, orders and delivery. From the Entity-relationship Diagram, all the table and attributes give their own meaning that which are primary key, foreign key, nullable, unique key. Primary key used to give references to the foreign key and is always unique where as foreign key always relates the data of primary key.

We insert different screenshots in database Model to show data store in different table. After that Data Relationship Diagram is created where we define how this all table works and give information, at first Suppliers table is created and then Items table is created because suppliers supplies Items and Items table contain primary key of suppliers as foreign key. Then Orders Table is created because Items have order, and then Customer table is created because Order makes Customer, here column of customer table give reference to Order table and finally Delivery table is created which delivered the items order given by Customer. Then Data dictionary is created which give information about which key is primary key, foreign key, unique key and nullable. At last we insert outputs of different queries which was written in this project queries, which gives exact data from database. The main Objective of this project is that all the students should have knowledge about queries and be more experiences in the field for overcoming the error while doing queries. At last on continuous research on different books and in different websites, I got more concept in the area of Queries. So from given coursework of Information System, many advantages were drawn having more knowledge and developing the habit of findings from individuals.

9. Bibliography:

Bibliography

unknown1, n.d. *Chapter 15 SQL Structured Query Language – Database Design.* [Online]
Available at: https://opentextbc.ca/.../sql-structured-quer...
[Accessed 19 07 2019].

Unknown2, n.d. *What is Data? - Math is Fun.* [Online]
Available at: https://www.mathsisfun.com/data/data.html
[Accessed 19 07 2019].

unknown3, n.d. *Relational Databases - 1&1 IONOS.* [Online]
Available at: https://www.ionos.com/.../relational-databas...
[Accessed 19 07 2019].

unknown4, n.d. *ERD is a tool to show - GATE Overflow.* [Online]
Available at: https://gateoverflow.in/.../erd-is-a-tool-to-sh..
[Accessed 19 07 2019].

Thank You