**This is a group project, but only one member will need to upload the sql file. You will post the file in Canvas with your group name. You need to complete the following:**

1. **Create tables in MySQL for the relational database schema you created in part two of your project**
2. **Create the relations between tables using “Relationships’ within the EER model diagram tool in MySQL (some of you have already done this)**
3. **Insert 5 records in each table**

* **You can also incorporate data from actual datasets, if you have identified a comparable dataset based on your database.**

1. **Create 7 queries in MySQL including:**

* **A query connecting two tables**
* **A query that contains two select statements and two tables**
* **A query that contains two select statements and three tables**
* **A query that addresses one of your descriptions outlined in project one, in part two**

**Response:**

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema github\_cable\_sales

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema github\_cable\_sales

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `github\_cable\_sales` DEFAULT CHARACTER SET utf8 ;

USE `github\_cable\_sales` ;

-- -----------------------------------------------------

-- Table `github\_cable\_sales`.`Region`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `github\_cable\_sales`.`Region` (

`head\_end` VARCHAR(5) NOT NULL,

`state` VARCHAR(2) NOT NULL,

`city` VARCHAR(45) NOT NULL,

`zipcode` VARCHAR(10) NOT NULL,

`comp\_code` VARCHAR(10) NOT NULL,

PRIMARY KEY (`head\_end`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `github\_cable\_sales`.`Customer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `github\_cable\_sales`.`Customer` (

`customer\_id` INT NOT NULL,

`last\_name` VARCHAR(45) NOT NULL,

`first\_name` VARCHAR(45) NOT NULL,

`user\_name` VARCHAR(45) NOT NULL,

`head\_end` VARCHAR(5) NOT NULL,

PRIMARY KEY (`customer\_id`),

INDEX `fk\_Customer\_Region1\_idx` (`head\_end` ASC) VISIBLE,

CONSTRAINT `fk\_Customer\_Region1`

FOREIGN KEY (`head\_end`)

REFERENCES `github\_cable\_sales`.`Region` (`head\_end`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `github\_cable\_sales`.`Plan`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `github\_cable\_sales`.`Plan` (

`plan\_id` INT NOT NULL,

`plan` VARCHAR(45) NOT NULL,

`plan\_desc` VARCHAR(45) NOT NULL,

PRIMARY KEY (`plan\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `github\_cable\_sales`.`Subscription`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `github\_cable\_sales`.`Subscription` (

`subscription\_id` INT NOT NULL,

`start\_date` DATE NOT NULL,

`offer\_code` VARCHAR(10) NOT NULL,

`customer\_id` INT NOT NULL,

`offer\_end\_date` DATE NULL,

`plan\_id` INT NOT NULL,

PRIMARY KEY (`subscription\_id`),

INDEX `fk\_Subscription\_Customer\_idx` (`customer\_id` ASC) VISIBLE,

INDEX `fk\_Subscription\_Plan\_idx` (`plan\_id` ASC) VISIBLE,

CONSTRAINT `fk\_Subscription\_Customer`

FOREIGN KEY (`customer\_id`)

REFERENCES `github\_cable\_sales`.`Customer` (`customer\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Subscription\_Plan`

FOREIGN KEY (`plan\_id`)

REFERENCES `github\_cable\_sales`.`Plan` (`plan\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `github\_cable\_sales`.`Billing`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `github\_cable\_sales`.`Billing` (

`account\_number` INT NOT NULL,

`account\_type` VARCHAR(45) NOT NULL,

`bill\_amount` DECIMAL(7,2) NOT NULL,

`balance\_due` DECIMAL(7,2) NOT NULL,

`billing\_date` DATE NOT NULL,

`customer\_id` INT NOT NULL,

`subscription\_id` INT NOT NULL,

PRIMARY KEY (`account\_number`),

INDEX `fk\_Billing\_Customer1\_idx` (`customer\_id` ASC) VISIBLE,

INDEX `fk\_Billing\_Subscription1\_idx` (`subscription\_id` ASC) VISIBLE,

CONSTRAINT `fk\_Billing\_Customer1`

FOREIGN KEY (`customer\_id`)

REFERENCES `github\_cable\_sales`.`Customer` (`customer\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Billing\_Subscription1`

FOREIGN KEY (`subscription\_id`)

REFERENCES `github\_cable\_sales`.`Subscription` (`subscription\_id`)

ON DELETE NO ACTION

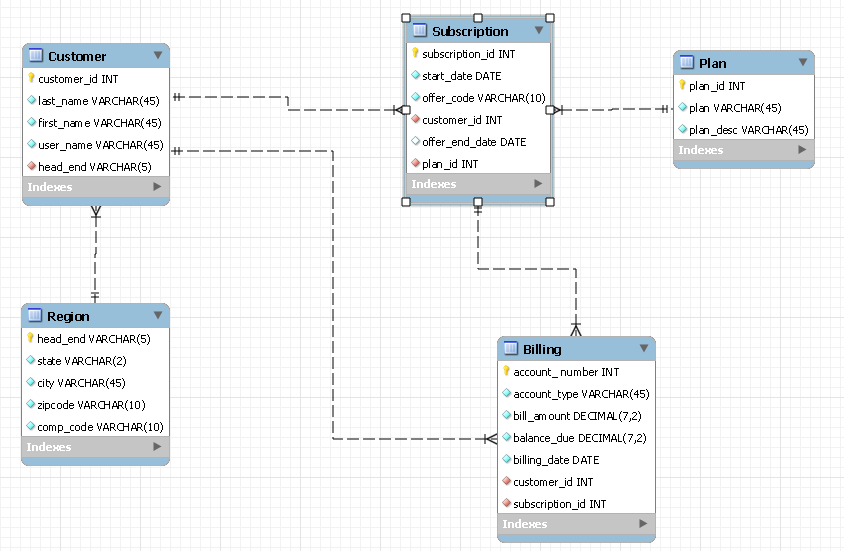
ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;



-- insert rows into the region table

INSERT INTO region(head\_end, state,city,zipcode,comp\_code)

VALUES

('YORK', 'NC', 'CHARLOTTE' ,'29745','ATT'),

('MECK', 'NC', 'CHARLOTTE' ,'28273','CHTR'),

('UNN', 'NC', 'CHARLOTTE' ,'28079','GOOG');

-- Group Project 2 insert rows into region table

INSERT INTO region(head\_end, state,city,zipcode,comp\_code)

VALUES

('GATN', 'NC', 'CHARLOTTE' ,'28006','ATT'),

('CBRS', 'NC', 'CHARLOTTE' ,'28025','CHTR');

-- insert rows into the Customer table

INSERT INTO Customer(customer\_id,last\_name,first\_name,user\_name,head\_end)

VALUES

(1001, 'Singaravel', 'Murali' ,'msinagara','YORK'),

(1002, 'Spaulding', 'Sydney' ,'sspauld5','MECK'),

(1003, 'Vellore', 'Krithika' ,'kvellore','MECK'),

(1004, 'Shaw', 'George' ,'gshaw','UNN');

-- Group Project 2 insert rows into customer table

INSERT INTO Customer(customer\_id,last\_name,first\_name,user\_name,head\_end)

VALUES

(1005, 'Jose', 'Mary' ,'msjose','GATN');

-- insert rows into the plan table

INSERT INTO plan(plan\_id,plan,plan\_desc)

VALUES

(101, 'Gold','unlimited internet 400 cable unlimited voice' ),

(201, 'Silver','50 GB internet 200 cable unlimited voice' ),

(301, 'Bronze','20 GB internet basic cable' );

-- Group Project 2 insert into plan table

INSERT INTO plan(plan\_id,plan,plan\_desc)

VALUES

(401, 'Basic I','10 GB internet basic cable' ),

(501, 'Basic Internet Only','10 GB internet only' ),

(601, 'Basic Cable Only','Basic cable' ),

(701, 'Basic Phone Only','Unlimited phone' );

-- insert rows into the subscription table

INSERT INTO subscription(subscription\_id,start\_date,offer\_code,customer\_id,offer\_end\_date,plan\_id)

VALUES

(1001, '2018-01-01','NEW', 1001, '2018-12-31',101),

(1002, '2018-06-01','CROSEL', 1004, '2018-12-31',101),

(1003, '2018-11-01','NA', 1002, NULL,301),

(1004, '2017-11-01','NEW', 1003, '2018-11-01',201);

-- Group Project 2 insert into subscription table

INSERT INTO subscription(subscription\_id,start\_date,offer\_code,customer\_id,offer\_end\_date,plan\_id)

VALUES

(1005, '2016-12-01','NEW', 1005, '2017-12-01',501),

(1006, '2016-12-01','NEW', 1005, '2017-12-01',601),

(1007, '2018-12-01','NA', 1002, NULL,701);

-- Group Project 2 inserts

-- insert rows into billing table

INSERT INTO billing(account\_number, account\_type, bill\_amount, balance\_due, billing\_date, customer\_id, subscription\_id)

VALUES

(100001, 'Residential','400.00','0.00','2018-11-01',1001,1001),

(100002, 'Residential','100.00','100.00','2018-11-15',1002,1003),

(100003, 'Residential','50.00','50.00','2018-12-01',1002,1007),

(100004, 'Residential','300.00','300.00','2018-11-01',1003,1004),

(100005, 'Residential','400.00','0.00','2018-11-01',1004,1002),

(100006, 'Residential','60.00','0.00','2018-11-01',1005,1005),

(100007, 'Residential','75.00','75.00','2018-11-15',1005,1006)

;

1. **Business Use case**: Get the list of customers and the dates on which they first started their subscription.

**Technical Use case:** A query connecting two tables.

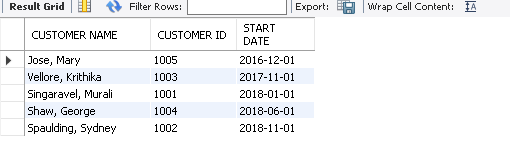
SELECT CONCAT(c.last\_name, ', ',c.first\_name) AS 'CUSTOMER NAME', c.customer\_id AS 'CUSTOMER ID', MIN(s.start\_date) AS 'START DATE'

FROM customer c

JOIN subscription s ON c.customer\_id= s.customer\_id

GROUP BY c.customer\_id

ORDER BY s.start\_date;



1. **Business Use case**: Find what plans are subscribed by most users, and see which subscriptions may need to be advertised more,

**Technical Use case:** A query that addresses one of the descriptions in project one and connecting two tables.

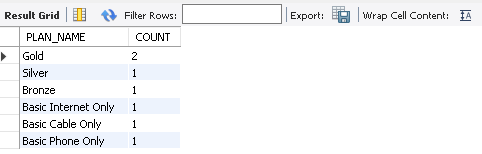
SELECT p.plan AS 'PLAN\_NAME',

COUNT(s.plan\_id) AS 'COUNT'

FROM subscription s

JOIN plan p USING (plan\_id)

GROUP BY plan\_id;



From the output data, it can be seen that plan ‘Gold’ is the most subscribed plan as compared to others.

1. **Business Use case**: Get the list of customers who are past their billing due date and have to be charged late penalty fees in the next billing invoice.

**Technical Use case:** A query that addresses one of the descriptions in project one and connecting two tables.

SELECT CONCAT(c.last\_name, ', ',c.first\_name) AS 'CUSTOMER NAME',

c.customer\_id AS 'CUSTOMER ID',

b.balance\_due AS 'BALANCE DUE',

b.billing\_date AS 'BILLING\_DATE'

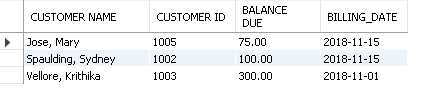
FROM customer c

JOIN billing b USING (customer\_id)

WHERE b.balance\_due > 0.00

AND b.billing\_date < current\_date()

ORDER BY c.last\_name;



1. **Business Use case:** Get the list of customers along with the competitors and whether the promotional offer has ended or not. This will give the list of customers who are likely to change their provider, with an assumption that a customer may churn at the end of the new customer signup offer

**Technical Use case:** A query that contains two select statements and three tables.

SELECT 'YES' AS'PROMO CUSTOMER' ,

CASE

WHEN s.offer\_end\_date < current\_date() THEN 'OFFER ENDED'

ELSE s.offer\_end\_date

END AS'OFFER\_END\_DATE',

CONCAT(c.last\_name, ', ',c.first\_name) AS 'CUSTOMER NAME',

r.comp\_code AS 'COMP CODE'

FROM subscription s

JOIN customer c USING (customer\_id)

JOIN region r USING (head\_end)

WHERE s.offer\_end\_date is NOT NULL

UNION ALL

SELECT 'NO' AS'PROMO CUSTOMER' ,

s.offer\_end\_date AS 'OFFER\_END\_DATE',

CONCAT(c.last\_name, ', ',c.first\_name) AS 'CUSTOMER NAME',

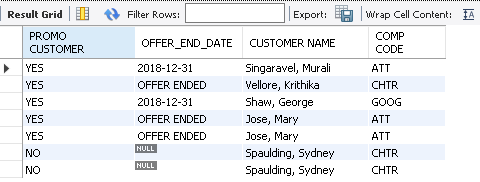
r.comp\_code AS 'COMP CODE'

FROM subscription s

JOIN customer c USING (customer\_id)

JOIN region r USING (head\_end)

WHERE s.offer\_end\_date is NULL;



From the output, we can see that there are 2 customers whose promotional offer has ended and there is competitor in their region which may increase their probability of churning. (NOTE: we have used UNION ALL because there are customers with multiple subscriptions which we wanted to include here).

1. **Business Use case:** Get the list of customers who have multiple subscriptions to see if a bundle plan can be advertised to these customers.

**Technical Use case:** A query that contains two select statements and two tables.

SELECT CONCAT(c.last\_name, ', ',c.first\_name) AS 'CUSTOMER NAME',

c.customer\_id AS 'CUSTOMER ID'

FROM customer c

WHERE c.customer\_id IN

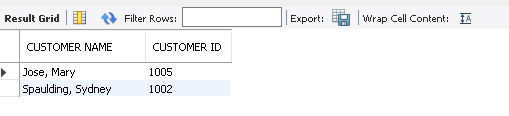
(SELECT customer\_id

FROM subscription

GROUP BY customer\_id

HAVING COUNT(\*) > 1)

ORDER BY c.last\_name;



1. **Business Use case:** Get the list of customers who are subscribed to just one product (Basic Internet Only or Basic Cable Only or Basic Phone Only) so that other products can be advertised for cross sell.

**Technical Use case:** A query connecting three tables.

SELECT CONCAT(c.last\_name, ', ',c.first\_name) AS 'CUSTOMER NAME',

c.customer\_id AS 'CUSTOMER ID',

p.plan AS 'PLAN'

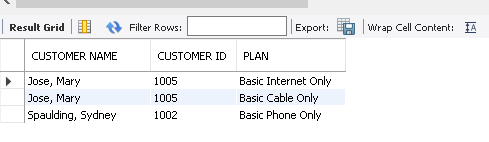
FROM customer c

JOIN subscription s USING (customer\_id)

JOIN plan p using (plan\_id)

WHERE s.plan\_id in (501,601,701)

ORDER BY c.last\_name;



1. **Business Use case:** Get the list of competitors in the customer region.

**Technical Use case:** A query that contains two select statements and two tables.

SELECT r.comp\_code, r.head\_end

FROM region r

WHERE r.head\_end IN

(SELECT DISTINCT(head\_end) FROM customer);

