Data Management and Business Intelligence - Assignment 1

Assignment Partners: Spanou Athina – p2821924 | Tziraki Maria - p2821930

Academic Year: 2019-2020 (Part-Time)

Section 1: Entity Relationship Diagram

The ERD model of the Telecom Provider consists of five distinct Entities, four Relationships and their corresponding Attributes. Specifically:

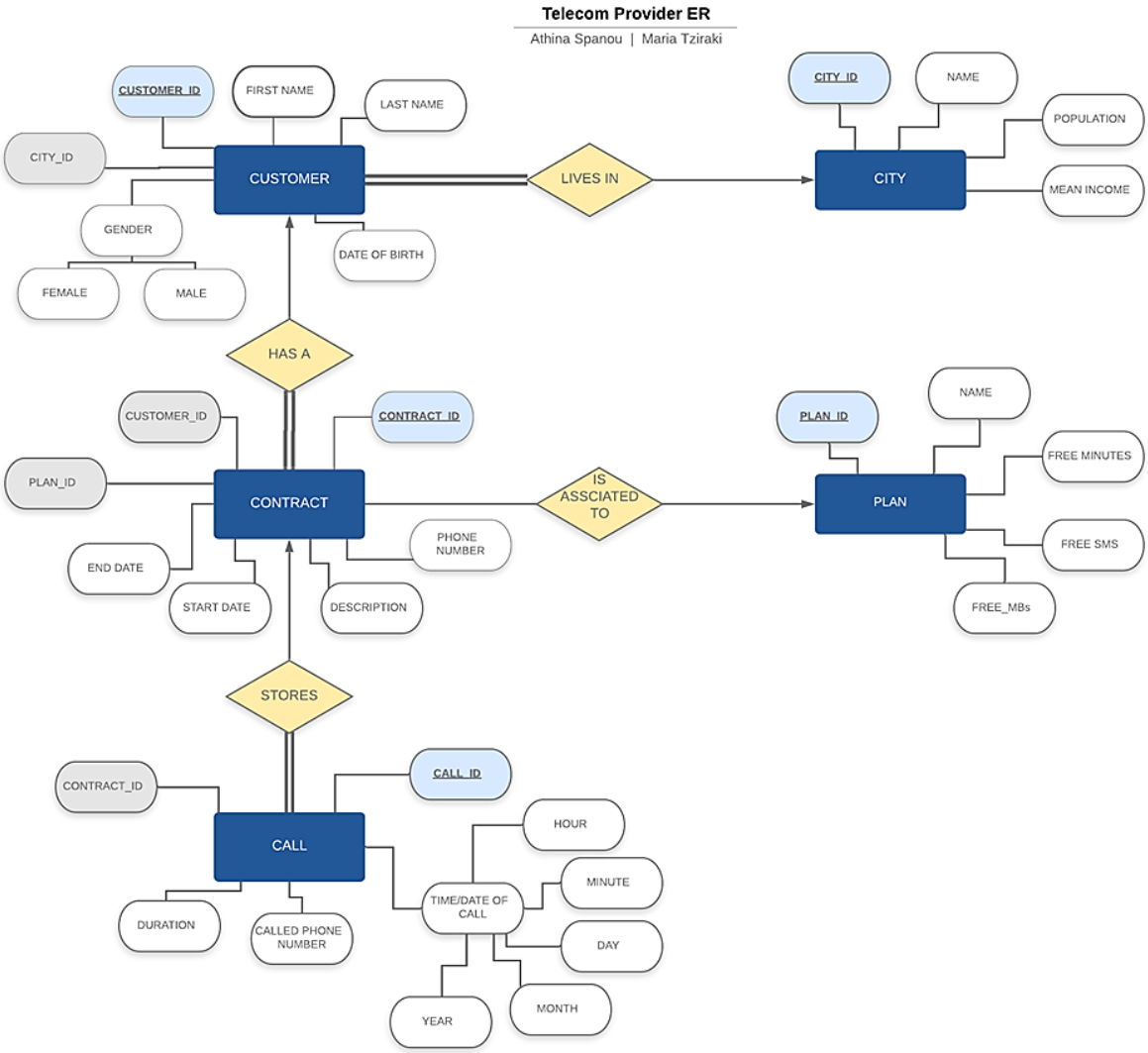
Entities: City, Customer, Contract, Plan, Call.

Entities are indicated with blue color.

Relationships are indicated with yellow color.

Primary Keys are indicated with light blue color.

Foreign Keys are indicated with light grey color.



Cardinalities:

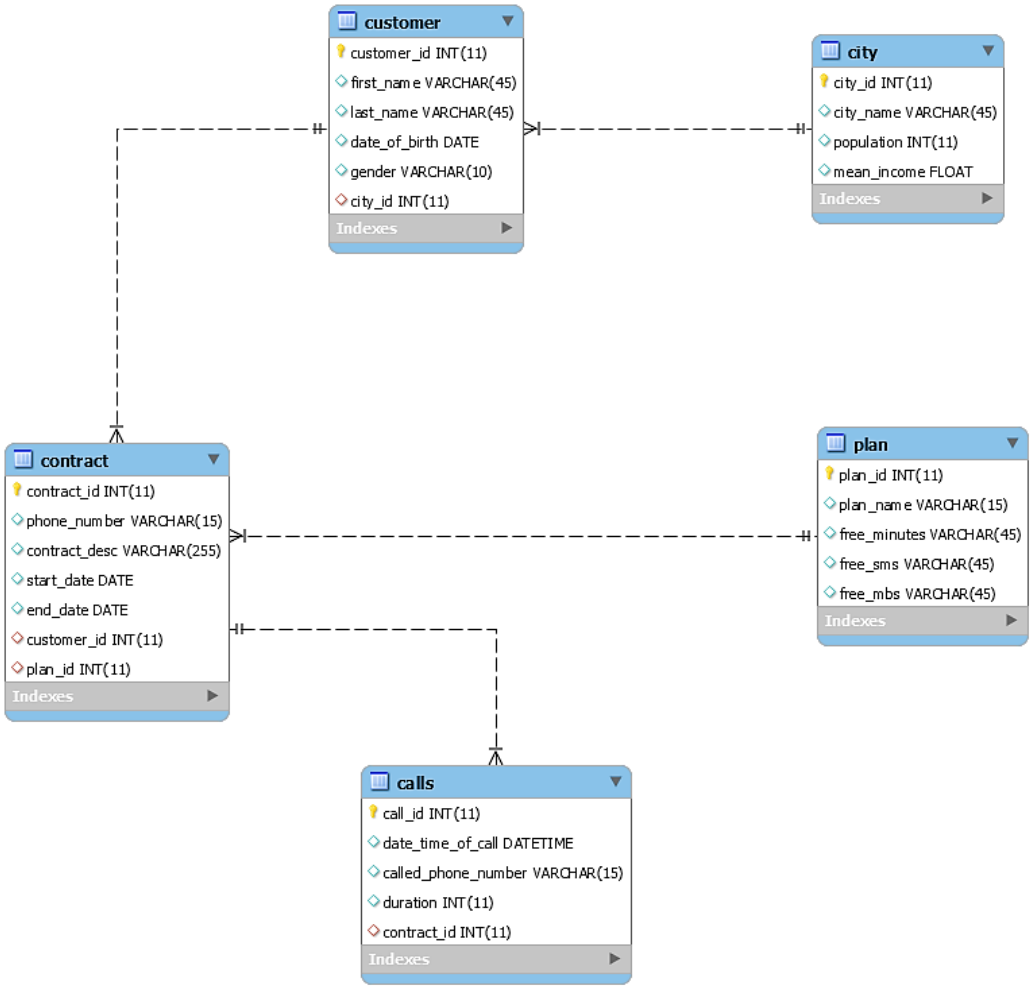
• Customer – City: many to one (∞ → 1). The entity Customer participates totally in the relationship as every Customer must live in a City.

• Customer - Contract: one to many (1 → ∞). The entity Contract participates totally in the relationship as every Contract must be about to a Customer.

• Contract - Plan: many to one (∞ → 1).

• Contract - Call: one to many (1 → ∞). The entity Call participates totally in the relationship as every Call must be made by a Contract.

Section 2: Relationship Schema Model



CREATE TABLE Statements.

```
CREATE DATABASE tp;

USE tp;

# First create tables that have no foreign keys

CREATE TABLE city (
city_id INT NOT NULL AUTO_INCREMENT,
city_name VARCHAR(45) NOT NULL,
population INT NOT NULL,
mean_income FLOAT NOT NULL CHECK (mean_income >= 0),
PRIMARY KEY (city_id)
);

CREATE TABLE customer (
customer_id INT NOT NULL AUTO_INCREMENT,
first_name VARCHAR(45) NOT NULL,
last_name VARCHAR(45) NOT NULL,
date_of_birth DATE NOT NULL,
gender VARCHAR(10) NOT NULL,
city_id INT NOT NULL,
PRIMARY KEY (customer_id),
FOREIGN KEY (city_id) REFERENCES city(city_id)
);

CREATE TABLE plan (
plan_id INT NOT NULL AUTO_INCREMENT,
plan_name VARCHAR(15) NOT NULL,
free_minutes VARCHAR(45) NOT NULL,
free_sms VARCHAR(45) NOT NULL,
free_mbs VARCHAR(45) NOT NULL,
PRIMARY KEY (plan_id)
);

CREATE TABLE contract (
contract_id INT NOT NULL AUTO_INCREMENT,
phone_number VARCHAR(15) NOT NULL,
contract_desc VARCHAR(255) NOT NULL,
start_date DATE NOT NULL,
end_date DATE NOT NULL,
customer_id INT NOT NULL,
plan_id INT NOT NULL,
PRIMARY KEY (contract_id),
FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
FOREIGN KEY (plan_id) REFERENCES plan(plan_id)
);
```

```
CREATE TABLE calls (
call_id INT NOT NULL AUTO_INCREMENT,
date_time_of_call DATETIME NOT NULL,
called_phone_number VARCHAR(15) NOT NULL,
duration INT NOT NULL,
contract_id INT NOT NULL,
PRIMARY KEY (call_id),
FOREIGN KEY (contract_id) REFERENCES contract(contract_id)
);
```

Insert a few records into the tables to test the queries.

```
INSERT INTO city(city_id,city_name,population,mean_income) VALUES
(1, 'Athens',664046,16587),
(2, 'Karpenisi', 13105, 5467),
(3, 'Heraklion', 312514, 12388),
(4, 'Nafplio', 14203, 4385),
(5, 'Corinth', 48132, 11965);
```

```
INSERT INTO customer(customer_id, first_name, last_name, date_of_birth, gender,
city_id) VALUES
(1,'Daphne', 'Dimitriou','1976/12/31','female',1),
(2,'Ioanna','Stavrou','1969/06/18','female',2),
(3,'Constantinos','Georgiou','1977/06/13','male',3),
(4,'Theodora','Papadopoulou','2000/12/25','female',4),
(5,'Demetrios','Theodorou','1991/02/13','male',5),
(6,'Gregorios','Economou','1992/07/16','male',1),
(7,'Markos','Petrides','1968/12/09','male',2),
(8,'Thalia','Vasiliou','1990/08/12','female',3);
```

```
INSERT INTO plan(plan_id, plan_name, free_minutes, free_sms, free_mbs) VALUES
(1,'Student',1200,1200,600),
(2,'Call+',600,100,500),
(3, 'Datalicious',500,500,2000),
(4, 'Mobile_Plus',2000,600,1000),
(5, 'Freedom',1000,600,800);
```

```
INSERT INTO contract(contract_id, phone_number, contract_desc, start_date,
end_date, customer_id, plan_id) VALUES
(1,'6945825551','personal','2017/05/24','2019/05/24',1,1),
(2,'6941627544','corporate','2018/01/18','2019/03/27',2,2),
(3,'6912345551','personal','2019/08/02','2020/08/02',3,3),
(4,'6974125558','corporate','2017/01/01','2019/12/15',4,4),
(5,'6945328879','personal','2018/12/18','2019/12/18',5,5),
(6,'6955324428','corporate','2019/12/18','2020/10/12',6,1),
(7,'6970326079','personal','2018/12/18','2019/11/28',7,2),
(8,'6969853264','personal','2018/12/18','2019/12/30',8,3),
(9,'6965987412','corporate','2018/12/18','2019/02/05',1,4),
(10,'6598745236','personal','2018/12/18','2019/04/08',2,5),
(11,'6958741236','corporate','2018/12/18','2019/12/08',3,1);
```

```
INSERT INTO calls(call_id, date_time_of_call, called_phone_number, duration,
contract_id) VALUES
(1,'2018/02/14 03:54','6970156809',72000,1),
(2,'2018/06/10 09:45','6974123214',18,2),
(3,'2017/10/09 22:10','6956321563',36000,3),
(4,'2017/02/14 09:38','6985236471',7500,4),
(5,'2018/06/10 09:55','6985236541',8,5),
(6,'2018/01/09 22:14','6974123214',42000,6),
(7,'2017/01/22 18:20','6951234785',1500,7),
(8,'2018/07/23 15:10','6987415236',38000,8),
(9,'2018/06/06 09:09','6985236541',14,9),
(10,'2017/07/17 11:11','6953628745',1530,10),
(11,'2018/10/09 22:20','6974123214',36000,1),
(12,'2018/06/20 10:10','6951236745',28,2),
(13,'2017/11/09 22:14','6974123214',4752,3),
(14,'2017/10/22 18:20','6958231473',42896,4),
(15,'2018/11/05 22:14','6914758239',2333,5),
(16, '2018/06/09 22:20','6974123214',36000,6),
(17,'2018/01/08 11:30','6985236541',33,7),
(18,'2017/07/12 19:10','6987415236',33000,8),
(19,'2018/01/07 04:10','6951234785',2500,9),
(20,'2018/02/14 14:14','6958231473',2365,11);
```

Section 3: SQL Code & Queries

a. Show the call id of all calls that were made between 8am and 10am on June 2018 having duration < 30

```
SELECT call_id AS 'Call ID'

FROM calls

WHERE MONTH(date_time_of_call) = 6 AND YEAR(date_time_of_call) = 2018 AND
HOUR(date_time_of_call) BETWEEN 8 AND 10 AND duration < 30 ;
```

Output:

Call ID
2
5
9
12

b. Show the first and last name of customers that live in a city with population greater than 20000

```
SELECT first_name as 'First Name', last_name as 'Last Name'

FROM customer, city

WHERE city.population > 20000 AND city.city_id = customer.city_id;
```

Output:

First Name	Last Name
Daphne	Dimitriou
Gregorios	Economou
Constantinos	Georgiou
Thalia	Vasiliou
Demetrios	Theodorou

c. Show the customer id that have a contract in the plan with name LIKE 'Freedom' (use nested queries).

```
SELECT customer.customer_id AS 'Customer ID'
FROM customer, contract

WHERE contract.plan_id IN (SELECT plan_id

                           FROM plan

                           WHERE plan_name LIKE '%Freedom%') AND
customer.customer_id=contract.customer_id;
```

Output:

Customer ID
5
2

d. For each contract that ends in less than sixty days from today, show the contract id, the phone number, the customer's id, his/her first name and his/her last name.

```
SELECT contract_id AS 'Contract ID', phone_number AS 'Phone Number',
customer.customer_id AS 'Customer ID', first_name AS 'First Name', last_name AS
'Last Name'
FROM customer, contract
WHERE DATEDIFF(CURRENT_TIMESTAMP, end_date) < 60 AND customer.customer_id =
contract.customer_id
GROUP BY contract_id;
```

Output:

Contract ID	Phone Number	Customer ID	First Name	Last Name
3	6912345551	3	Constantinos	Georgiou
11	6958741236	3	Constantinos	Georgiou
4	6974125558	4	Theodora	Papadopoulou
5	6945328879	5	Demetrios	Theodorou
6	6955324428	6	Gregorios	Economou
7	6970326079	7	Markos	Petrides
8	6969853264	8	Thalia	Vasiliou

e. For each contract id and each month of 2018, show the average duration of calls

```
SELECT contract_id AS 'Contract ID', ROUND(AVG(duration),2) AS 'Month',
AVG(duration) AS 'Average Duration of Calls'
FROM calls
WHERE YEAR(date_time_of_call) = 2018
GROUP BY contract_id, MONTH(date_time_of_call)
ORDER BY contract_id ASC;
```

Output:

Contract ID	Month	Average Duration of Calls
1	2	72000.00
1	10	36000.00
2	6	23.00
5	6	8.00
5	11	2333.00
6	1	42000.00
6	6	36000.00
7	1	33.00
8	7	38000.00
9	1	2500.00
9	6	14.00
11	2	2365.00

f. Show the total duration of calls in 2018 per plan id

```
SELECT plan.plan_id AS 'Plan ID', SUM(calls.duration) AS 'Total Duration of Calls'
FROM plan, contract, calls
WHERE plan.plan_id = contract.plan_id AND contract.contract_id = calls.contract_id
AND YEAR(date_time_of_call) = 2018
GROUP BY plan.plan_id;
```

Output:

Plan ID	Total Duration of Calls
1	188365
2	79
3	38000
4	2514
5	2341

g. Show the top called number among TP's customers in 2018

```
SELECT called_phone_number AS 'Top Called Phone Number', COUNT(called_phone_number)
AS 'Times Called'
FROM calls
WHERE YEAR(date_time_of_call) = 2018
GROUP BY called_phone_number
ORDER BY COUNT(called_phone_number) DESC
LIMIT 1;
```

Output:

	Top Called Phone Number	Times Called
▶	6974123214	4

h. Show the contract ids and the months where the total duration of the calls was greater than the free minutes offered by the plan of the contract

```
#Create View in order to merge tables calls and contract and keep the necessary
#information


CREATE VIEW total_calls(contract, call_month, duration,plan) AS

SELECT contract.contract_id, MONTH(date_time_of_call), (duration), contract.plan_id
FROM calls
INNER JOIN contract ON (calls.contract_id=contract.contract_id);

SELECT total_calls.contract AS 'Contract ID', total_calls.call_month AS 'Month',
total_calls.duration AS 'Duration', free_minutes AS 'Free Minutes'
FROM total_calls, plan
WHERE total_calls.plan=plan.plan_id
GROUP BY total_calls.contract,total_calls.call_month
HAVING SUM(total_calls.duration)>(free_minutes*60);
```

Output:

	Contract ID	Month	Duration	Free Minutes
▶	3	10	36000	500
	8	7	38000	500

i. For each month of 2018, show the percentage change of the total duration of calls compared to the same month of 2017

```
SELECT Month2018 AS 'Month', concat(round((duration2018-
duration2017)/duration2017*100),'%') AS 'Percent Change'
FROM ( SELECT SUM(duration) AS duration2018, MONTH(date_time_of_call) AS Month2018
       FROM calls
       WHERE YEAR(date_time_of_call)=2018
       GROUP BY Month2018) AS calls_2018,
       (SELECT SUM(duration) AS duration2017, month(date_time_of_call) AS Month2017
        FROM calls
        WHERE YEAR(date_time_of_call)=2017
        GROUP BY Month2017) AS calls_2017
WHERE calls_2018.Month2018= calls_2017_2017.Month2017;
```

Output:

	Month	Percent Change
▶	2	892%
	1	2869%
	7	10%
	10	-54%
	11	-51%

j. For each city id and calls made in 2018, show the average call duration by females and the average call duration by males (i.e. three columns)

```
CREATE VIEW total_duration(callid,duration, crontractid,customerid) AS
SELECT call_id, duration, calls.contract_id, contract.customer_id
FROM calls
JOIN contract ON calls.contract_id
WHERE YEAR(date_time_of_call)=2018 AND calls.contract_id=contract.contract_id;

CREATE VIEW female(city_id,femaleduration) AS
SELECT city_id, AVG(duration)
FROM customer, total_duration
WHERE gender LIKE 'female' AND customer.customer_id=total_duration.customerid
GROUP BY city_id
ORDER BY city_id;

CREATE VIEW male(city_id,maleduration) AS
SELECT city_id, AVG(duration)
FROM customer, total_duration
WHERE gender LIKE 'male' AND customer.customer_id=total_duration.customerid
GROUP BY city_id
ORDER BY city_id;

(SELECT city_id AS 'City ID', ROUND(femaleduration,2)AS 'Average Duration of
Females', ROUND(maleduration,2)AS 'Average Duration of Males'
FROM female LEFT JOIN male USING (city_id))
UNION
SELECT city_id, ROUND(femaleduration,2), ROUND(maleduration,2)
FROM male LEFT JOIN female USING (city_id);
```

Output:

	City ID	Average Duration of Females	Average Duration of Males
▶	1	27628.50	39000.00
	2	23.00	33.00
	3	38000.00	2365.00
	5	NULL	1170.50

k. For each city id, show the city id, the ratio of the total duration of the calls made from customers staying in that city in 2018 over the total duration of all calls made in 2018, and the ratio of the city's population over the total population of all cities (i.e three columns)

```
CREATE VIEW city_population(city_id,population_ratio) AS

SELECT city_a.city_id, city_a.population/sum(city_b.population)

FROM city AS city_a JOIN city AS city_b

GROUP BY city_a.city_id;


CREATE VIEW city_call(city_id,calls_ratio) AS

SELECT city.city_id, SUM(a.duration)/(SELECT SUM(duration) FROM calls WHERE
YEAR(date_time_of_call)=2018) as duration_ratio

FROM city JOIN customer ON customer.city_id=city.city_id JOIN contract ON
contract.customer_id=customer.customer_id JOIN calls AS a ON
a.contract_id=contract.contract_id WHERE YEAR(date_time_of_call)=2018

GROUP BY city.city_id;


(SELECT city_id AS 'City ID', ROUND(population_ratio,2) AS 'Population Ratio',
ROUND(calls_ratio,2) AS 'Calls Ratio'

FROM city_population LEFT JOIN city_call USING (city_id))

UNION

SELECT city_id, ROUND(population_ratio,2), ROUND(calls_ratio,2)

FROM city_call LEFT JOIN city_population USING (city_id);
```

Output:

City ID	Population Ratio	Calls Ratio
1	0.63	0.82
2	0.01	0.00
3	0.30	0.17
4	0.01	NULL
5	0.05	0.01

Section 4: Query k using R

By trying to connect MySQL with R, we encountered the following problem:

"Problem loading the "caching_sha2_password" plugin, due to the default settings of the newest MySQL version."

This problem was solved with the following method:

Open console > connect to MySQL > Run:

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'newrootpassword';

4. Using the programming language of your choice, connect to the database and implement query (k) – without using GROUP BY SQL statements

```
#install.packages('RMySQL')

library(RMySQL)

#You need to input your own database name, username and password here

mydb <- dbConnect(MySQL(), user='root', password='athinaMysql833@', dbname='tp',
host='127.0.0.1')


rs <- dbSendQuery(mydb, "select * from contract")
contract <-fetch(rs,n=-1)
contract
str(contract)

rs <- dbSendQuery(mydb, "select * from customer")
customer <-fetch(rs,n=-1)
customer
str(customer)

query <- paste("select * from calls where year(date_time_of_call)=2018")
sendquery <- dbSendQuery(mydb, query)
calls <- fetch(sendquery, n=-1)

call_contract <- merge(calls,contract)
call_contract
call_contr_cust <- merge(call_contract,customer)
call_contr_cust


total_dur_per_city <-
aggregate(x=list(total_duration=call_contr_cust$duration),by=list(city_id=call_cont
r_cust$city_id), FUN=sum)

total_dur_per_city


total_dur_2018 <- sum(calls$duration);total_dur_2018


call_ratio<-
data.frame(total_dur_per_city$city_id,total_dur_per_city$total_duration/total_dur_2
018)
names(call_ratio)<-c("city","calls_ratio"); call_ratio
```

```
rs<-dbSendQuery(mydb, "select * from city")
city <- fetch(rs,n=-1)
city
str(city)

total_population<-sum(city$population); total_population


city_population<- data.frame(city$city_id,city$population/total_population)
names(city_population)<-c("city","population_ratio"); city_population


final<-merge(x = round(city_population,2), y = round(call_ratio,2), by = "city",
all.x = TRUE);
names(final) <- c("City ID","Population Ratio","Calls Ratio"); final
```

Output:

	City ID	Population Ratio	Calls Ratio
1	1	0.63	0.82
2	2	0.01	0.00
3	3	0.30	0.17
4	4	0.01	NA
5	5	0.05	0.01