|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| VILNIAUS KOLEGIJA  UNIVERSITY OF APPLIED SCIENCES  FACULTY OF ELECTRONICS AND INFORMATICS  Image result for viko logo | | |  | | | VILNIUS COLLEGE  Image result for viko logoFACULTY OF ELECTRONICS AND INFORMATICS |
|  | | |  | | |  |
| **MOBILE NETWORKS DATABASE** | | |  | | | **INTRODUCTION TO INFORMATICS** |
| REPORT  6531BX028 PI18E | | |  | | | PRACTICAL ASSIGNMENT  SPOTIFY USER MANUAL  6531BX028 PI18E |
| STUDENT | DŽIUGAS PEČIULEVIČIUS | STUDENT | | DŽIUGAS PEČIULEVIČIUS |
| (SIGNATURE)  6/5/2020 | | |  | | | LECTURER |
| LECTURER | JUSTINAS ZAILSKAS | (SIGNATURE)  10/17/2018 | | VIRGILIJUS KUKLIERIUS |
| (SIGNATURE)  6/5/2020 | | |  | | | 2018 |

2020

Table of Contents

[INTRODUCTION 3](#_Toc42360810)

[PRACTICAL ASSIGNMENT ANALYSIS 4](#_Toc42360811)

[**1.** **ERD diagram** 4](#_Toc42360812)

[**2.** **Relational model** 4](#_Toc42360813)

[**3.** **Normalization steps** 5](#_Toc42360814)

[**4.** **Database and Constructed Queries** 6](#_Toc42360815)

[**1.** **CREATE DATABASE command** 6](#_Toc42360816)

[**2.** **CREATE TABLE command** 6](#_Toc42360817)

[**3.** **SELECT & DESCRIBE commands** 7](#_Toc42360818)

[**4.** **INSERT commands** 8](#_Toc42360819)

[**5.** **Other queries** 9](#_Toc42360820)

[CONCLUSIONS 22](#_Toc42360821)

[RESOURCES 22](#_Toc42360822)

# INTRODUCTION

**Practice problem**

With the emergence of new technologies companies often encounter a problem of using an outdated database. The problem introduced in this document is how to remodel the old database into one that incorporates all additional services provided by the company.

**Short description**

Firstly, our task is to analyze all of the company’s website that provides mobile network service. Then we need to make an ERD and Relational model then construct a database and create 20 queries for the database.

**Goal**

The goal is to provide a practical solution of the problem at hand, using knowledge of database design and provide a visualization of the functionality of the new database.

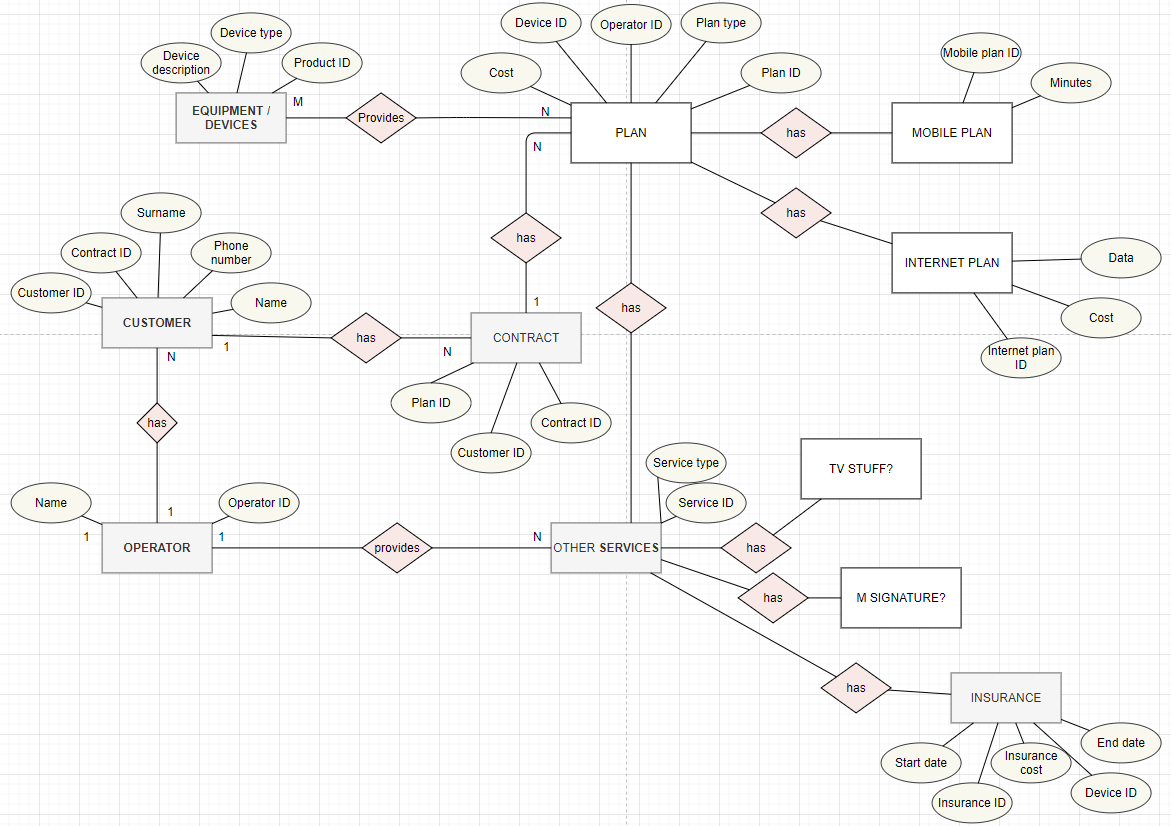
**My database file here:**

# <https://drive.google.com/file/d/1CQHw1E4pOKsA7yUA64ah00jU_d0ELATw/view?usp=sharing>

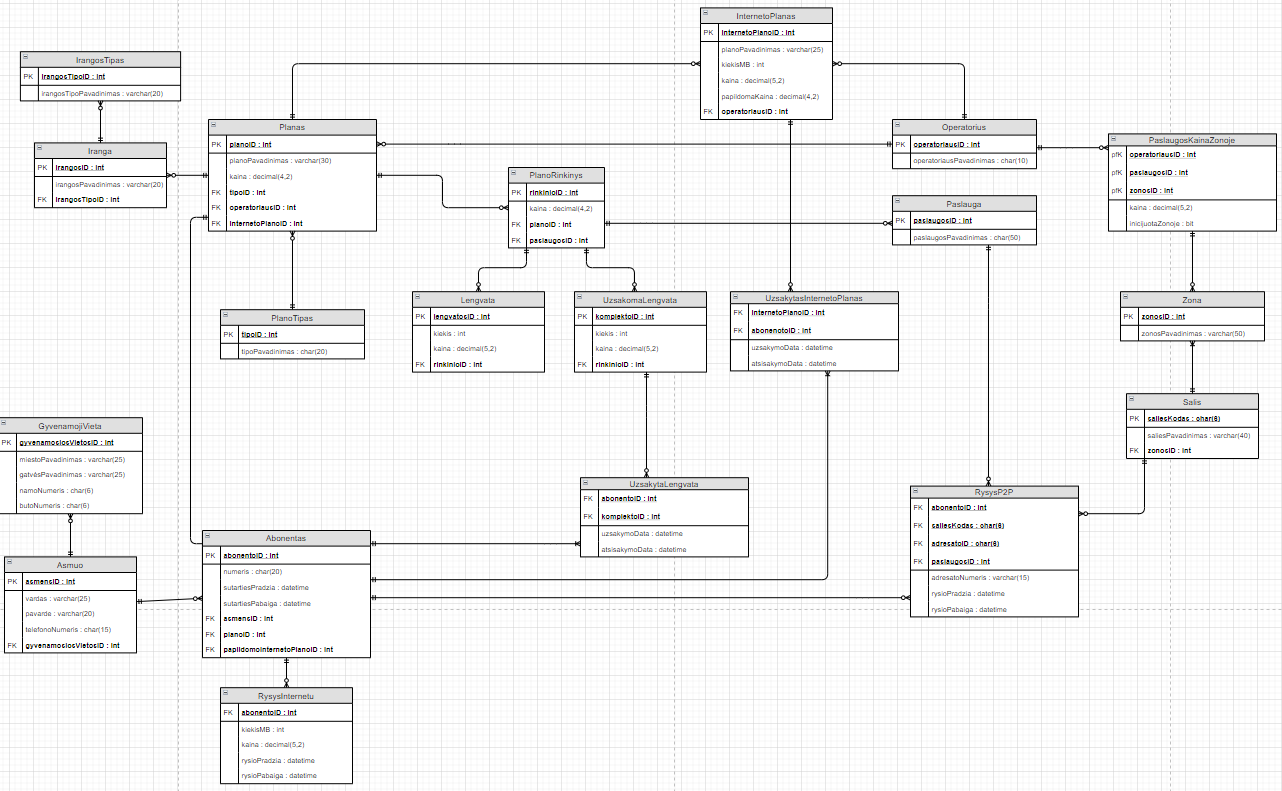
# 

# PRACTICAL ASSIGNMENT ANALYSIS

## **ERD diagram**

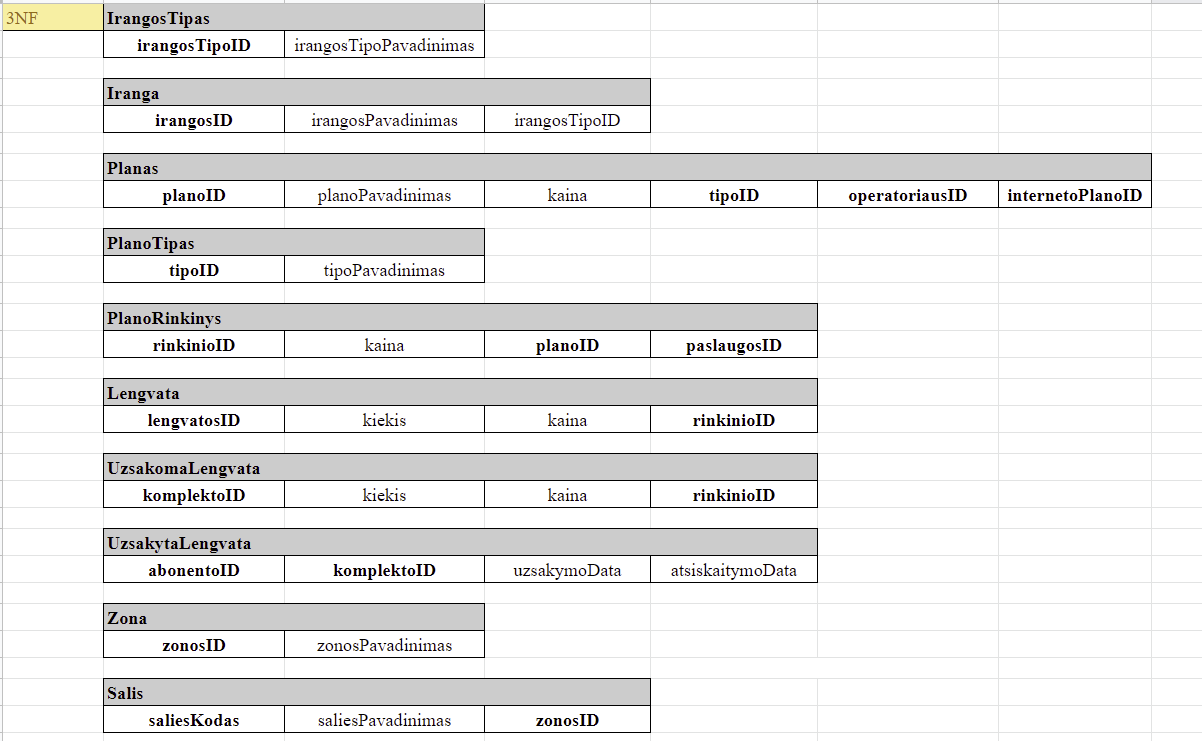


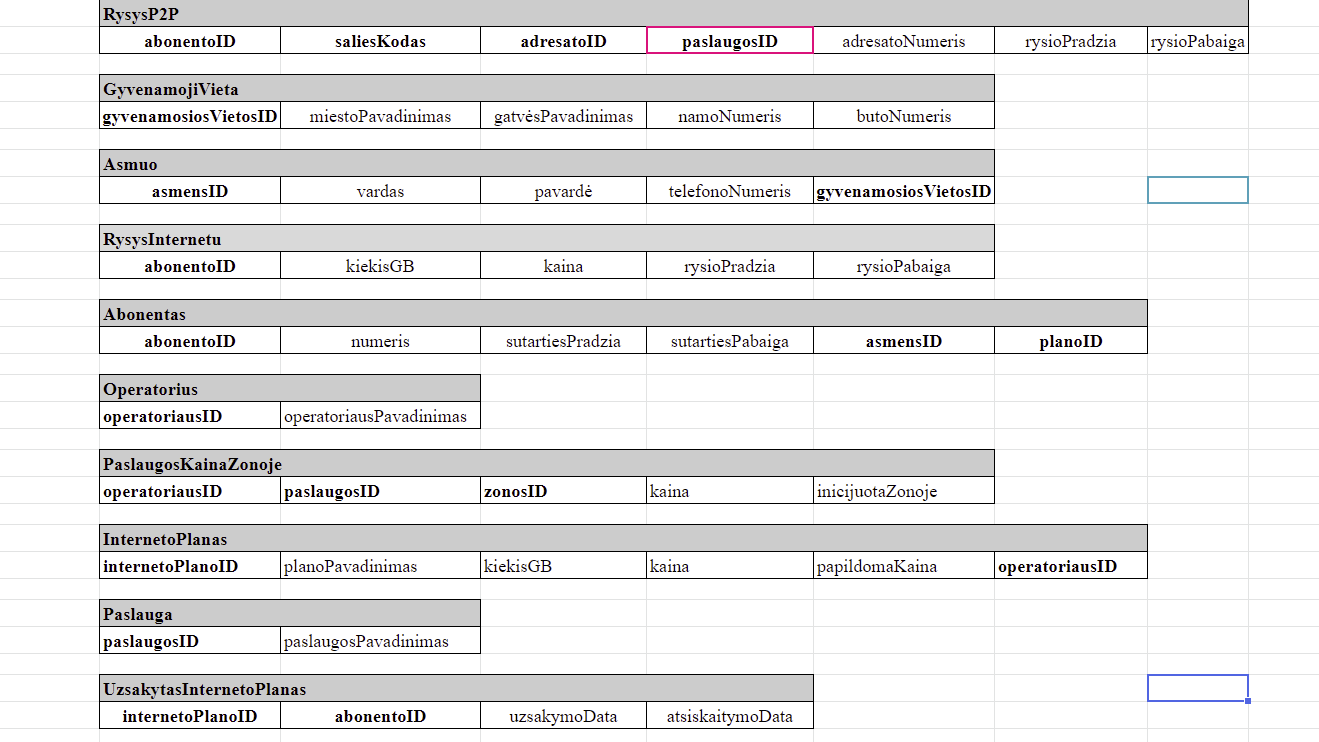
## **Relational model**



## **Normalization steps**

All the table where in 3rd form already there wasn’t a need to apply normalization steps





## **Database and Constructed Queries**

### **CREATE DATABASE command**

The database was written in MySQL Workbench.

To create a database, CREATE DATABASE command was used:



### **CREATE TABLE command**

After creating a database, we need to create tables, so later on we can insert all the necessary information.

CREATE TABLE IrangosTipas (

irangosTipoID INTEGER NOT NULL AUTO\_INCREMENT,

irangosTipoPavadinimas VARCHAR(20) NOT NULL UNIQUE,

PRIMARY KEY(irangosTipoID)

);

SELECT \* FROM IrangosTipas;

DESCRIBE IrangosTipas;

CREATE TABLE Iranga (

irangosID INTEGER NOT NULL AUTO\_INCREMENT,

irangosPavadinimas VARCHAR(20) NOT NULL,

irangosTipoPavadinimas VARCHAR(20) NOT NULL,

PRIMARY KEY(irangosID),

FOREIGN KEY(irangosTipoPavadinimas) REFERENCESIrangosTipas(irangosTipoPavadinimas)

);

SELECT \* FROM Iranga;

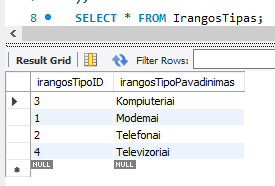
DESCRIBE Iranga;

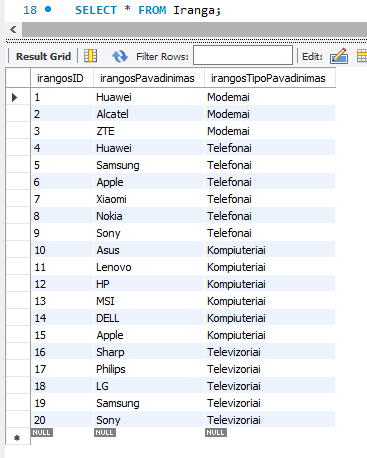
On queries above we see three types of commands. Firstly, we create our first table using CREATE TABLE command and call it *IrangosTipas*. This table should describe which type of devices are available. *IrangosTipas* has type ID and a name. Type ID is a primary key (PK) for this table.

Then table *Iranga* was created and it stores information about actual device. And it is connected to a *IrangosTipas* so it has a foreign key (FK) of device type name. And *Iranga* also has an ID and a name. *IrangosID* is a PK and *irangosTipoPavadinimas* is a FK of a table.

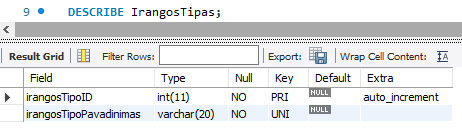
### **SELECT & DESCRIBE commands**

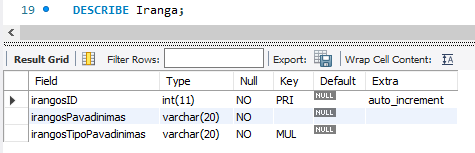
Then we can see other two queries in the picture. The first is a SELECT \* query which select and displays all the data which is stored in the selected tables.





Then we have a DESCRIBE table keyword. This keyword displays types of data which is stored in the table.





### **INSERT commands**

-- irangosTipoID -> AUTO\_INCREMENT

INSERT INTO IrangosTipas(irangosTipoPavadinimas) VALUES ('Modemai');

INSERT INTO IrangosTipas(irangosTipoPavadinimas) VALUES ('Telefonai');

INSERT INTO IrangosTipas(irangosTipoPavadinimas) VALUES ('Kompiuteriai');

INSERT INTO IrangosTipas(irangosTipoPavadinimas) VALUES ('Televizoriai');

-- irangosID -> AUTO\_INCREMENT

INSERT INTO Iranga(irangosPavadinimas, irangosTipoPavadinimas) VALUES ('Huawei', 'Modemai');

INSERT INTO Iranga(irangosPavadinimas, irangosTipoPavadinimas) VALUES ('Alcatel', 'Modemai');

INSERT INTO Iranga(irangosPavadinimas, irangosTipoPavadinimas) VALUES ('ZTE', 'Modemai');

INSERT INTO Iranga(irangosPavadinimas, irangosTipoPavadinimas) VALUES ('Huawei', 'Telefonai');

INSERT INTO Iranga(irangosPavadinimas, irangosTipoPavadinimas) VALUES ('Samsung', 'Telefonai');

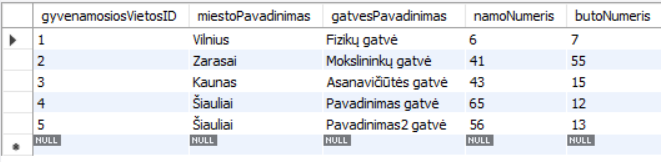
INSERT INTO Iranga(irangosPavadinimas, irangosTipoPavadinimas) VALUES ('Asus', 'Kompiuteriai');

### **Other queries**

***-- 1. OUTPUTS THE COUNT OF ENTRIES IN THE TABLE "GYVENAMOJI VIETA"***

SELECT COUNT(\*) FROM GyvenamojiVieta;

This below is what we have in our *GyvenamojiVieta* table, we can see that there are 5 entries in the database.



We can now see how many entries we have in our database by making a command mentioned above.



***-- 2. OUTPUTS INFORMATION ABOUT DEVICE NAME AND WHAT TYPE IT BELONGS TO***

SELECT Iranga.irangosID , Iranga.irangosPavadinimas, IrangosTipas.irangosTipoID, IrangosTipas.irangosTipoPavadinimas

FROM Iranga

INNER JOIN IrangosTipas

ON Iranga.irangosTipoPavadinimas = IrangosTipas.irangosTipoPavadinimas;



Basically we selected (SELECT) necessary columns that we want to display device and device type IDs, and a device name and its type FROM table *Iranga* and we use INNER JOIN to join both tables and display information (ON) where table *Iranga* has same comlumn name as *IrangosTipas*

***-- 3. OUTPUTS USERS COUNTS IN EACH CITY (DOES NOT OUTPUT CITIES THAT DO NOT HAVE ANY USERS)***

SELECT GyvenamojiVieta.gyvenamosiosVietosID, miestoPavadinimas, COUNT(asmensID)

FROM GyvenamojiVieta

INNER JOIN Asmuo ON GyvenamojiVieta.gyvenamosiosVietosID= Asmuo.gyvenamosiosVietosID

GROUP BY GyvenamojiVieta.gyvenamosiosVietosID, miestoPavadinimas

ORDER BY COUNT(asmensID) DESC;



In this query we output users count in each city from highest user count to lowest. We target two tables: *GyvenamojiVieta* and *Asmuo*. From GyvenamojiVieta we join with Asmuo where their gyvenamosiosVietosID match. Then we group the values and list them from biggest to smallest.

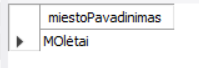
It doesn’t list cities that have zero users in them. Should use LEFT JOIN if we want to see cities that don’t have any users in them.

***-- 4. OUTPUTS CITIES THAT DO NOT HAVE A SINGLE USER IN THAT CITY***

SELECT GyvenamojiVieta.miestoPavadinimas FROM GyvenamojiVieta

WHERE gyvenamosiosVietosID NOT IN (SELECT gyvenamosiosVietosID FROM Asmuo);

We SELECT table column *miestoPavadinimas* FROM table *GyvenamojiVieta* to select a city name to show. Then we specify that we want to select IDs only WHERE *gyvenamosiosVietosID* is not in table *Asmuo*.



There was a small typo in the name of the city. Instead of a capital O there should be a small one. But these tables show city names that don’t have any users in them.

***-- 5. OUTPUTS ALL COMPUTERS***

SELECT Iranga.irangosID, Iranga.irangosPavadinimas, IrangosTipas.irangosTipoPavadinimas

FROM Iranga

INNER JOIN IrangosTipas

ON Iranga.irangosTipoPavadinimas =IrangosTipas.irangosTipoPavadinimas

WHERE IrangosTipas.irangosTipoPavadinimas = 'Kompiuteriai';



With this query a goal was to display device ID, type name and a device name. So I selected two tables: *Iranga* and *IrangosTipas.* So firstly we select values from each tables we want to display: from table *Iranga* deviceID and device name and from *IrangosTipas* we select device type name. Then we join both tables using INNER JOIN on deviceTypeName and then we output information where keyword matches the keyword “*Kompiuteriai*”.

***-- 6. OUTPUTS OPERATORS NAME WHO HAS TO OFFER THE MOST AMMOUNT OF PLANS***

SELECT operatoriausPavadinimas, COUNT(Planas.operatoriausID) AS planuSkaicius

FROM Operatorius INNER JOIN Planas ON Operatorius.operatoriausID = Planas.operatoriausID

GROUP BY operatoriausPavadinimas

ORDER BY COUNT(planuSkaicius) DESC

LIMIT 1;

In this table we select the operator which has most plans to offer.



***-- 7. OUTPUTS ALL OPERATORS OFFERED PLANS THAT HAS A PRICE AVERAVE OVER 15 EUROS. BASICALLY IT OUTPUTS OPERATOR AND ITS AVERAGE PRICE. (ALL IN DESCENDING ORDER)***

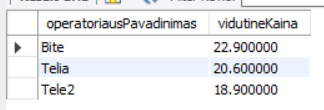
SELECT operatoriausPavadinimas, AVG(kaina) AS vidutineKaina

FROM Planas INNER JOIN Operatorius ON Operatorius.operatoriausID = Planas.operatoriausID

WHERE kaina > 15

GROUP BY operatoriausPavadinimas

ORDER BY vidutineKaina DESC;



***-- 8. OUTPUTS PLANS THAT HAS A PRICE OF LESS THAN 10 EUROS***

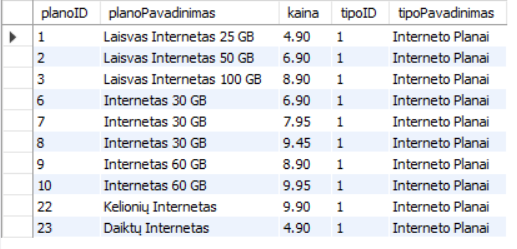
SELECT Planas.planoID, Planas.planoPavadinimas,Planas.kaina, PlanoTipas.tipoID,PlanoTipas.tipoPavadinimas

FROM Planas

INNER JOIN PlanoTipas

ON Planas.tipoID = PlanoTipas.tipoID

WHERE Planas.kaina <= 10;



***-- 9. CLIENT AND SUBSCRIBER WITH PLAN INFORMATION***

SELECT Asmuo.vardas, Asmuo.pavarde, Abonentas.abonentoID, Abonentas.numeris,Planas.planoPavadinimas,Planas.kaina

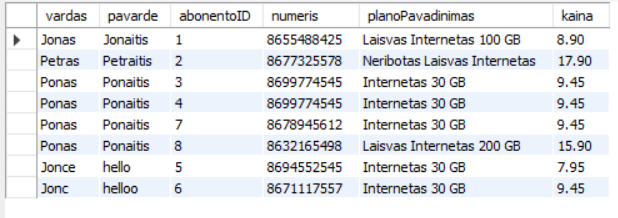
FROM Asmuo

INNER JOIN Abonentas

ON Asmuo.asmensID = Abonentas.asmensID

INNER JOIN Planas

ON Abonentas.planoID = Planas.planoID;



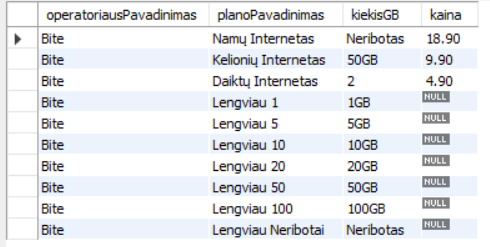
***-- 10. OPERATORS OFFERED INTERNET PLANS***

SELECT Operatorius.operatoriausPavadinimas, InternetoPlanas.planoPavadinimas, InternetoPlanas.kiekisGB, InternetoPlanas.kaina

FROM Operatorius

INNER JOIN InternetoPlanas

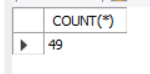
ON Operatorius.operatoriausID = InternetoPlanas.operatoriausID;



NULL prices are the internet plans that come with phone plans and its all inside the phone bill.

***-- 11. ALL OFFERED PLANS COUNT***

SELECT COUNT(\*) FROM Planas;



***-- 12. OUTPUTS ALL INFORMATION THAT WAS SEARCHED BY NAME***

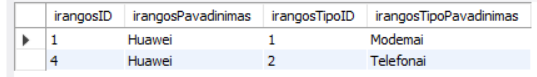
SELECT Iranga.irangosID , Iranga.irangosPavadinimas, IrangosTipas.irangosTipoID, IrangosTipas.irangosTipoPavadinimas

FROM Iranga

INNER JOIN IrangosTipas

ON Iranga.irangosTipoPavadinimas =IrangosTipas.irangosTipoPavadinimas

WHERE Iranga.irangosPavadinimas= 'Huawei';



***-- 13. OUTPUTS PHONE PLANS***

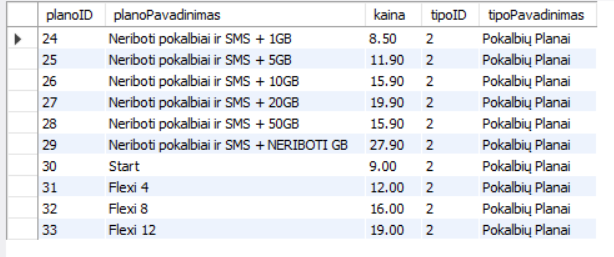
SELECT Planas.planoID, Planas.planoPavadinimas,Planas.kaina, PlanoTipas.tipoID,PlanoTipas.tipoPavadinimas

FROM Planas

INNER JOIN PlanoTipas

ON Planas.tipoID = PlanoTipas.tipoID

WHERE PlanoTipas.tipoID = 2;



***-- 14. OUTPUTS INTERNET PLANS***

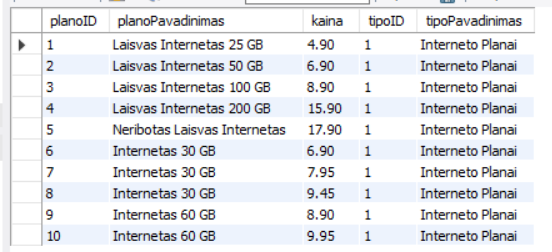
SELECT Planas.planoID, Planas.planoPavadinimas,Planas.kaina, PlanoTipas.tipoID,PlanoTipas.tipoPavadinimas

FROM Planas

INNER JOIN PlanoTipas

ON Planas.tipoID = PlanoTipas.tipoID

WHERE PlanoTipas.tipoID = 1;



***-- 15. OUTPUTS ALL PLANS THAT HAVE A NAME “FLEXI”***

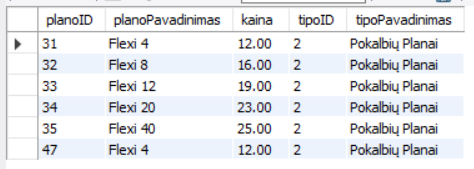
SELECT Planas.planoID, Planas.planoPavadinimas,Planas.kaina, PlanoTipas.tipoID,PlanoTipas.tipoPavadinimas

FROM Planas

INNER JOIN PlanoTipas

ON Planas.tipoID = PlanoTipas.tipoID

WHERE Planas.planoPavadinimas LIKE 'Flexi%';



***-- 16. OUTPUTS ALL PLANS WITH THE NAME “LENGVIAU”***

SELECT Planas.planoID, Planas.planoPavadinimas,Planas.kaina, PlanoTipas.tipoID,PlanoTipas.tipoPavadinimas

FROM Planas

INNER JOIN PlanoTipas

ON Planas.tipoID = PlanoTipas.tipoID

WHERE Planas.planoPavadinimas LIKE 'LENGVIAU%';



***-- 17. ZONE AND SERVICE FEES RELATION***

SELECT Zona.zonosPavadinimas, PaslaugosKainaZonoje.kaina, Paslauga.paslaugosPavadinimas

FROM Zona

INNER JOIN PaslaugosKainaZonoje

ON Zona.zonosID=PaslaugosKainaZonoje.zonosID

INNER JOIN Paslauga

ON PaslaugosKainaZonoje.paslaugosID = Paslauga.paslaugosID;



***-- 18. USER AND SUBSCRIBTION WITH INTERNET CONNECTIONS INFORMATION***

SELECT Asmuo.vardas, Asmuo.pavarde, Abonentas.abonentoID, Abonentas.numeris, Abonentas.sutartiesPradzia, Abonentas.sutartiesPabaiga, RysysInternetu.kiekisGB, RysysInternetu.kaina

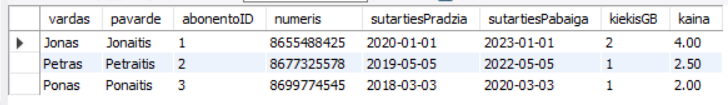
FROM Asmuo

INNER JOIN Abonentas

ON Asmuo.asmensID= Abonentas.asmensID

INNER JOIN RysysInternetu

ON Abonentas.abonentoID = RysysInternetu.abonentoID;



***-- 19. OUTPUTS USERS INFORMATION AND USERS LIVING LOCATION***

SELECT Asmuo.asmensID, Asmuo.vardas, Asmuo.pavarde, Asmuo.telefonoNumeris, GyvenamojiVieta.miestoPavadinimas, GyvenamojiVieta.gatvesPavadinimas,GyvenamojiVieta.namoNumeris, GyvenamojiVieta.butoNumeris

FROM Asmuo

INNER JOIN GyvenamojiVieta

ON Asmuo.gyvenamosiosVietosID =GyvenamojiVieta.gyvenamosiosVietosID;



***-- 20. OUTPUTS USERS PLAN AND INTERNET PLAN INFORMTION***

SELECT Asmuo.vardas, Asmuo.pavarde, Abonentas.abonentoID, Abonentas.numeris,Planas.planoPavadinimas,Planas.kaina, InternetoPlanas.kiekisGB

FROM Asmuo

INNER JOIN Abonentas

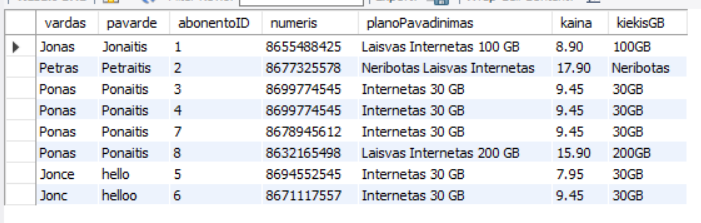
ON Asmuo.asmensID= Abonentas.asmensID

INNER JOIN Planas

ON Abonentas.planoID = Planas.planoID

INNER JOIN InternetoPlanas

ON Planas.internetoPlanoID = InternetoPlanas.internetoPlanoID;



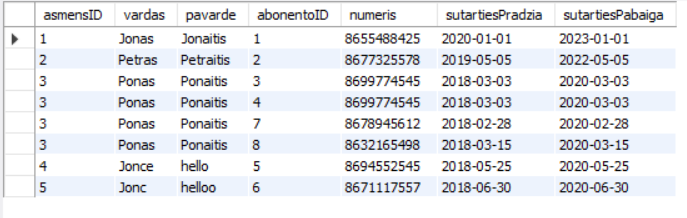
***-- 21. OUTPUTS USERS NAME, SURNAME AND PLAN INFORMATION***

SELECT Asmuo.asmensID, Asmuo.vardas, Asmuo.pavarde, Abonentas.abonentoID, Abonentas.numeris, Abonentas.sutartiesPradzia, Abonentas.sutartiesPabaiga

FROM Asmuo

INNER JOIN Abonentas

ON Asmuo.asmensID = Abonentas.asmensID;



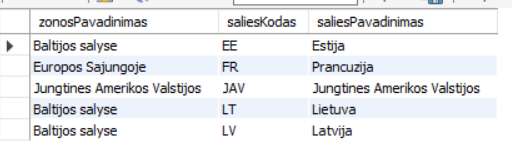
***-- 22. ZONES AND COUNTRIES INFORMATION***

SELECT Zona.zonosPavadinimas,Salis.saliesKodas, Salis.saliesPavadinimas

FROM Zona

INNER JOIN Salis

ON Zona.zonosID = Salis.zonosID;



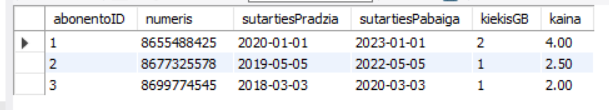
***-- 23. OUTPUTS SUBSCRIBERS INFORMATION AND ASSOCIATION WITH INTERNET CONNECTION INFORMATION***

SELECT Abonentas.abonentoID, Abonentas.numeris, Abonentas.sutartiesPradzia, Abonentas.sutartiesPabaiga, RysysInternetu.kiekisGB, RysysInternetu.kaina

FROM Abonentas

INNER JOIN RysysInternetu

ON Abonentas.abonentoID = RysysInternetu.abonentoID;



# CONCLUSIONS

The websites of companies that provide mobile network services were analyzed, to figure out what technologies and services are being offered these days. From the collected data an ERD and relational model have been constructed to visually show the design of the new database. After that, queries for the database creation and insertion have been written. And, retrieval of data has been written in a variety of queries that show functionality that the database provides.

# RESOURCES

Tele2: <https://tele2.lt/privatiems/>

Telia: <https://www.telia.lt/privatiems>

Bite: <https://www.bite.lt/>