MSc Business Analytics

Data Management & Business Intelligence

SQL and Relational Databases

MESOLORA STAMATOULA-GERASIMOULA
Registration number: f2822308
e-mail: sta.mesolora@aueb.gr

Contents

ASSIGNMENT DESCRIPTION

The aim of this assignment is to develop a relational database for "TelcoX", a telecom provider company, in order to monitor customers, calls and plans. Using mySQL, a relational database management system, there have been created tables which describe and organize the company's data such as existing contracts, calls that have been made and the plans associated with them.

First of all, there has been designed an Entity-Relationship Diagram (ERD) to help us understand all relationships, attributes and necessary constraints between entities and it is accompanied by a relational schema showing the development of the tables and the connection between their contents, where it exists. Then some records have been inserted into the tables in order to check their proper functioning and use them to perform the requested queries.

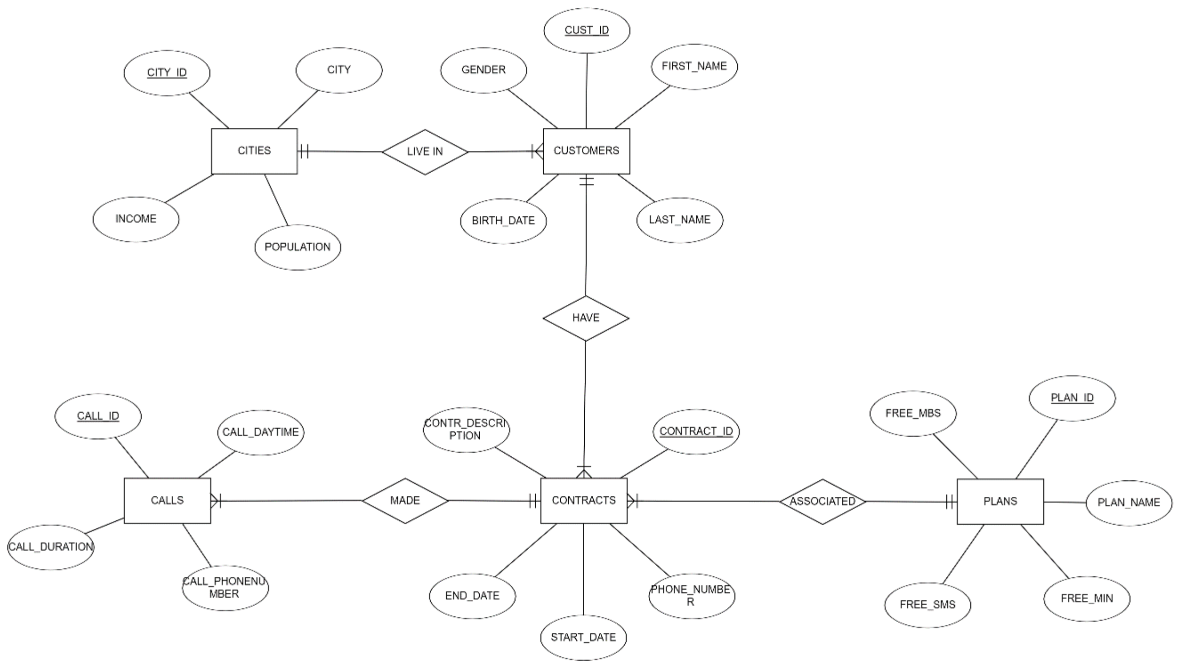
Finally, all the queries were carried out using SQL code and the last one was also carried out using the Python programming language.

DELIVERABLE 1

Entity-Relationship Diagram (ERD)

The Entity-Relationship Diagram (ERD) has been created through "ERDPlus" (https://erdplus.com/), an online database modeling tool.

Entity-Relationship Diagram



The above diagram shows the entities, their attributes, including primary keys, and the way they are related.

Rectangles represent entities:

- CUSTOMERS
- CITIES
- CONTRACTS
- PLANS
- CALLS

Rhombus represent the relationship between entities:

- Customers **live in** Cities
- Customers **have** Contracts
- Every Contract is **associated** with a Plan
- Calls are **made** from phone numbers which are included to Contracts

Ellipeses represent entities' attributes (underlined attributes correspond to primary keys):

- CUST_ID, FIRST_NAME, LAST_NAME, BIRTH_DATE, GENDER, and CITY for CUSTOMERS, with CUST_ID as primary key.
- CITY_ID, CITY, POPULATION, INCOME for CITIES, with CITY_ID as primary key
- PLAN_ID, PLAN_NAME, FREE_MIN, FREE_SMS, FREE_MBS for PLANS, with PLAN_ID as primary key
- CONTRACT_ID, PHONE_NUMBER, START_DATE, END_DATE, CONTR_DESCRIPTION, PLAN_ID, CUST_ID for CONTRACTS, with CONTRACT_ID as primary key
- CALL_ID, CALL_DAYTIME, CALL_PHONENUMBER, CALL_DURATION, CONTRACT_ID for CALLS, with CALL_ID as primary key

lines connect traits to entities, entities to relationships and vice versa. The notation in every line indicates the interaction between the related entities.

- Each customer can only live in one city, but one city can host up to one customers
- One customer may be assigned to many contracts, but every contract is related to only one customer
- Calls are made by a phone number that belongs to a contract and every phone number can be related to many calls.

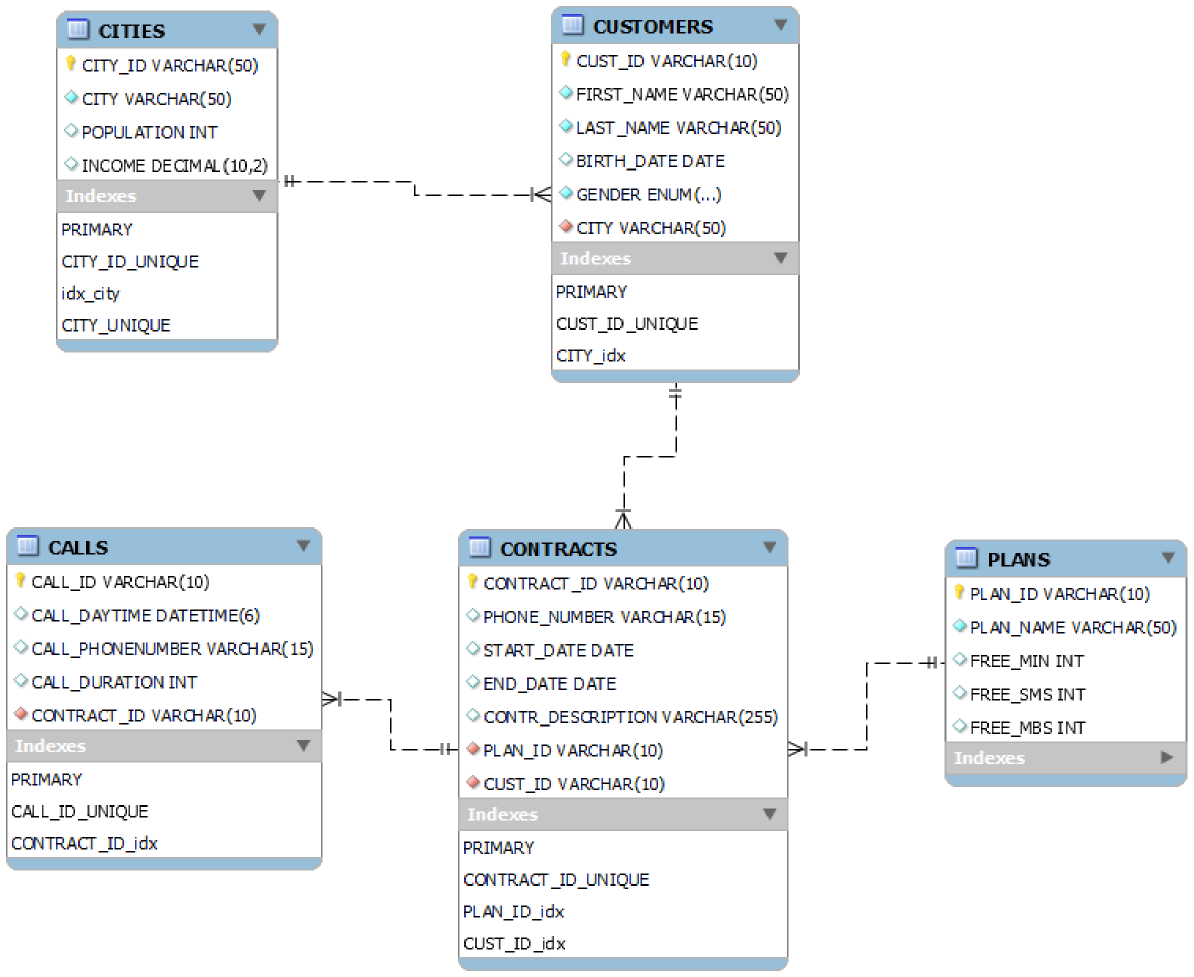
DELIVERABLE 2

Relational Model

The Relational Schema has been created in mySQL and consists of the entities, attributes and relationships that were presented above. Unlike ERD, the relational schema is able to configurate foreign keys, allowing us to see in greater detail the interaction between entities and their attributes.

In the first part, there have been created the tables that will be included to the schema with specification to the type of their values and their keys (primary & foreign).

<u>Relational Model</u>



<u>Step by step description:</u>

➢ Table CITIES:
CITY_ID is the primary key of the table and CITY has been set to index in order to achieve connection with CUSTOMERS table.

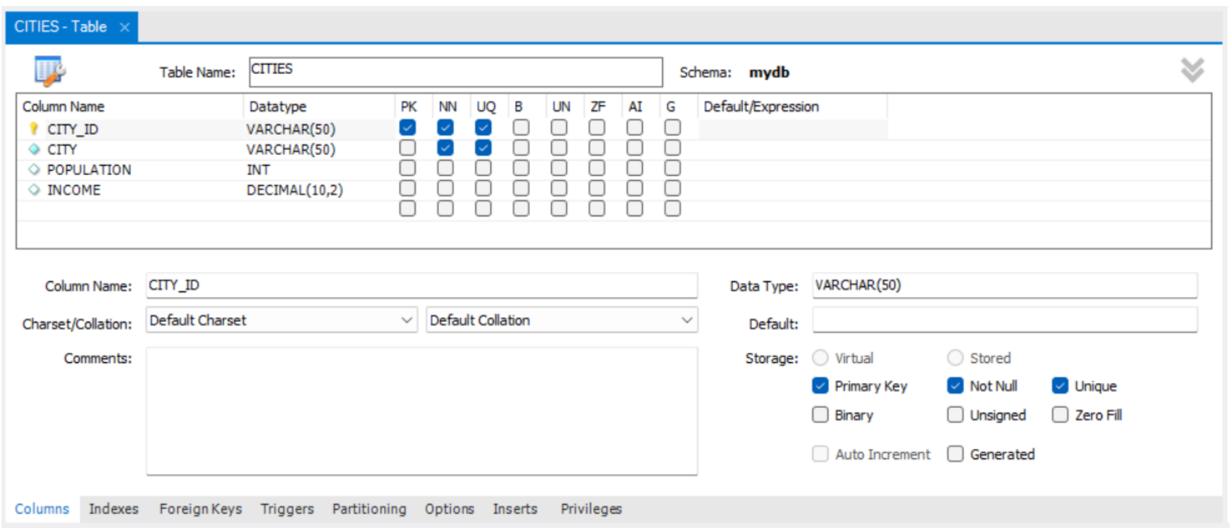

CITIES - Table

Table Name: CITIES Schema: mydb

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
CITY_ID	VARCHAR(50)	✓	✓	✓						
CITY	VARCHAR(50)		✓	✓						
POPULATION	INT									
INCOME	DECIMAL(10,2)									

Column Name: CITY_ID
Data Type: VARCHAR(50)
Charset/Collation: Default Charset / Default Collation
Default:
Comments:
Storage: Virtual / Stored
[✓ Primary Key] [✓ Not Null] [✓ Unique]
Binary / Unsigned / Zero Fill
Auto Increment / Generated

Columns Indexes Foreign Keys Triggers Partitioning Options Inserts Privileges

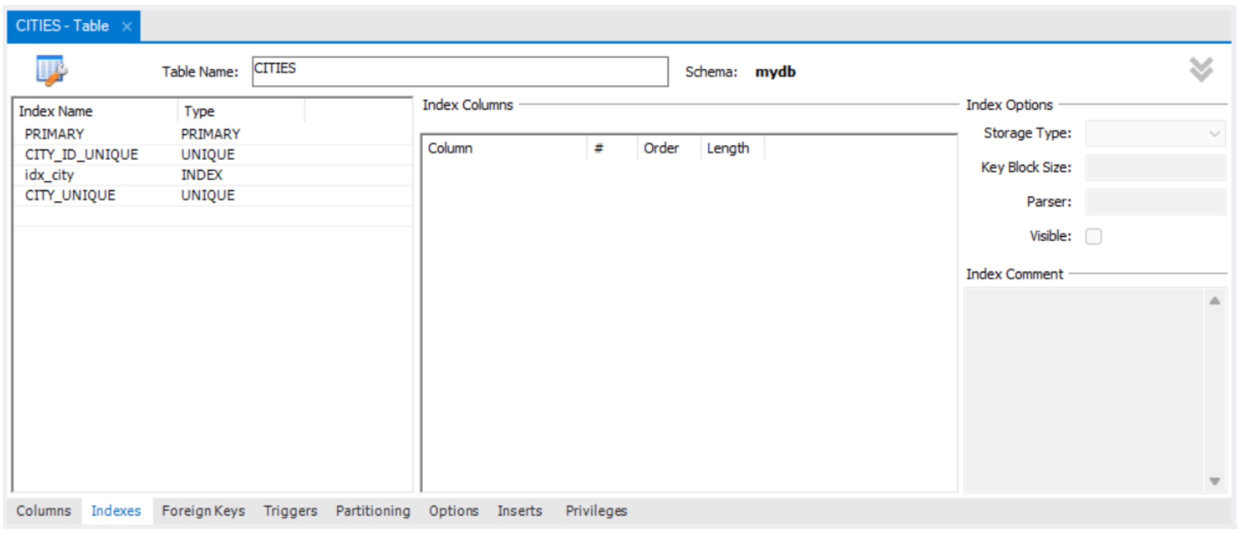


Table Name: CITIES Schema: **mydb**

Index Name	Type
PRIMARY	PRIMARY
CITY_ID_UNIQUE	UNIQUE
idx_city	INDEX
CITY_UNIQUE	UNIQUE

Column	#	Order	Length

- ➢ Table CUSTOMERS:
  CUST_ID is the primary key of the table and CITY is a foreign key from table CITIES.

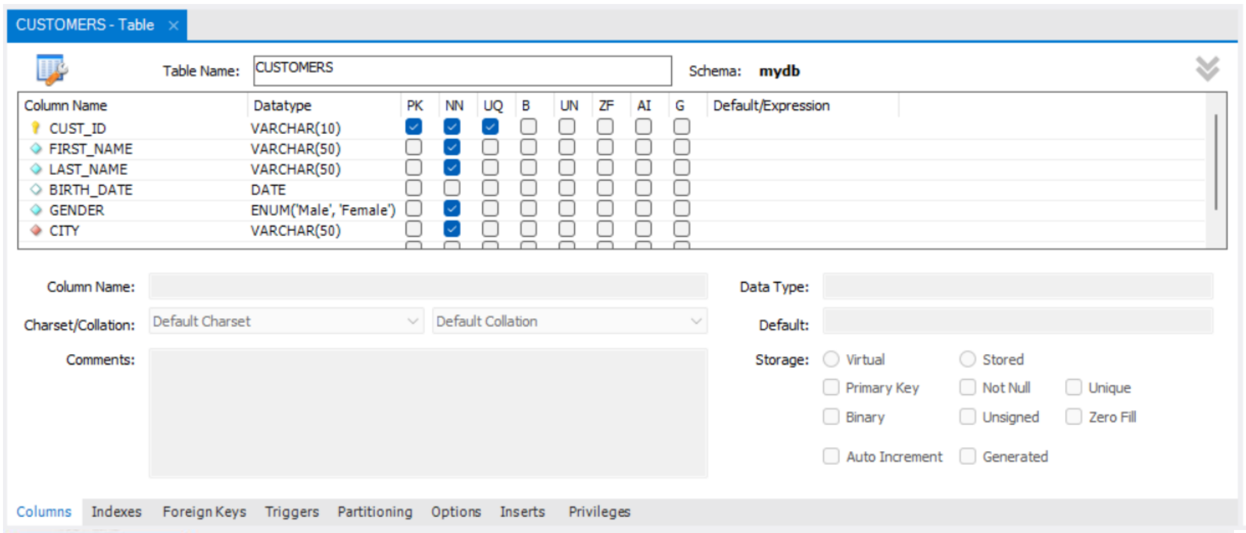


Table Name: CUSTOMERS Schema: **mydb**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
CUST_ID	VARCHAR(10)	✓	✓	✓						
FIRST_NAME	VARCHAR(50)		✓							
LAST_NAME	VARCHAR(50)		✓							
BIRTH_DATE	DATE									
GENDER	ENUM('Male', 'Female')		✓							
CITY	VARCHAR(50)		✓							

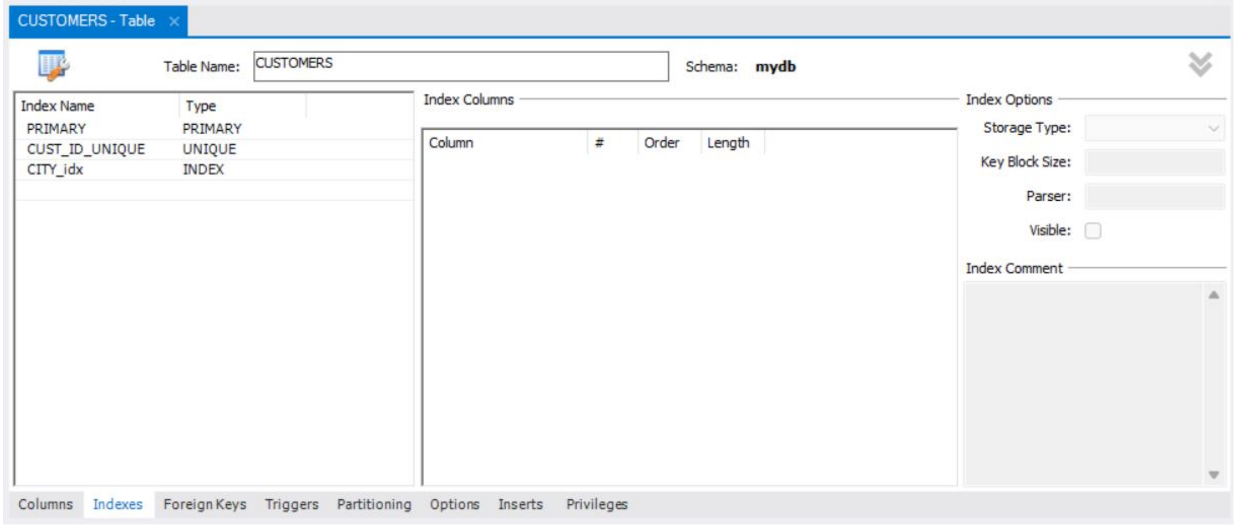
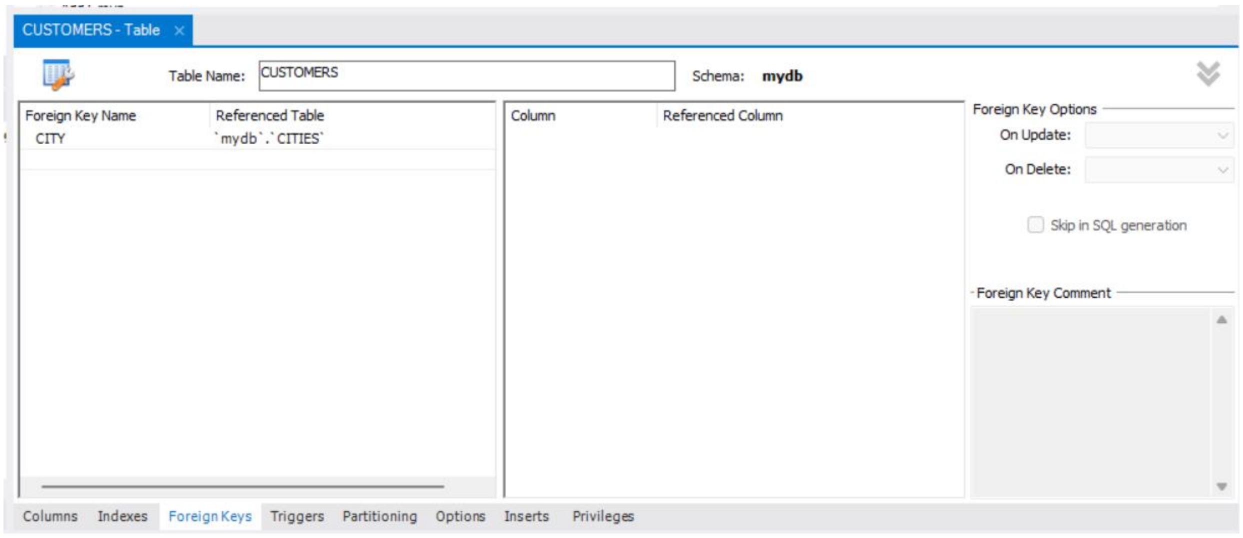


Table Name: CUSTOMERS Schema: **mydb**

Index Name	Type
PRIMARY	PRIMARY
CUST_ID_UNIQUE	UNIQUE
CITY_idx	INDEX

Column	#	Order	Length



CUSTOMERS - Table

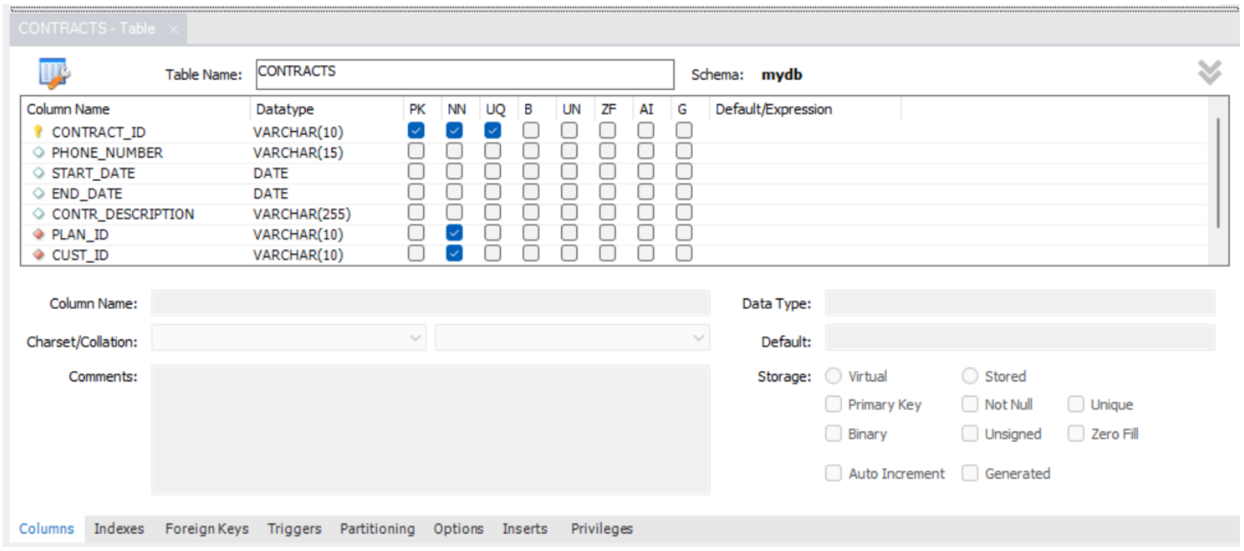
Table Name: CUSTOMERS Schema: mydb

Foreign Key Name	Referenced Table
CITY	\`mydb\`.\`CITIES\`

Column	Referenced Column

Foreign Key Options
On Update:
On Delete:
Skip in SQL generation
Foreign Key Comment

➢ Table CONTRACTS:
CONTRACT_ID is the primary key of the table and PLAN_ID and CUST_ID are foreign keys from tables PLANS and CUSTOMERS respectively.



CONTRACTS - Table

Table Name: CONTRACTS Schema: mydb

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
CONTRACT_ID	VARCHAR(10)	✓	✓	✓						
PHONE_NUMBER	VARCHAR(15)									
START_DATE	DATE									
END_DATE	DATE									
CONTR_DESCRIPTION	VARCHAR(255)									
PLAN_ID	VARCHAR(10)		✓							
CUST_ID	VARCHAR(10)		✓							

Column Name:
Charset/Collation:
Comments:
Data Type:
Default:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated

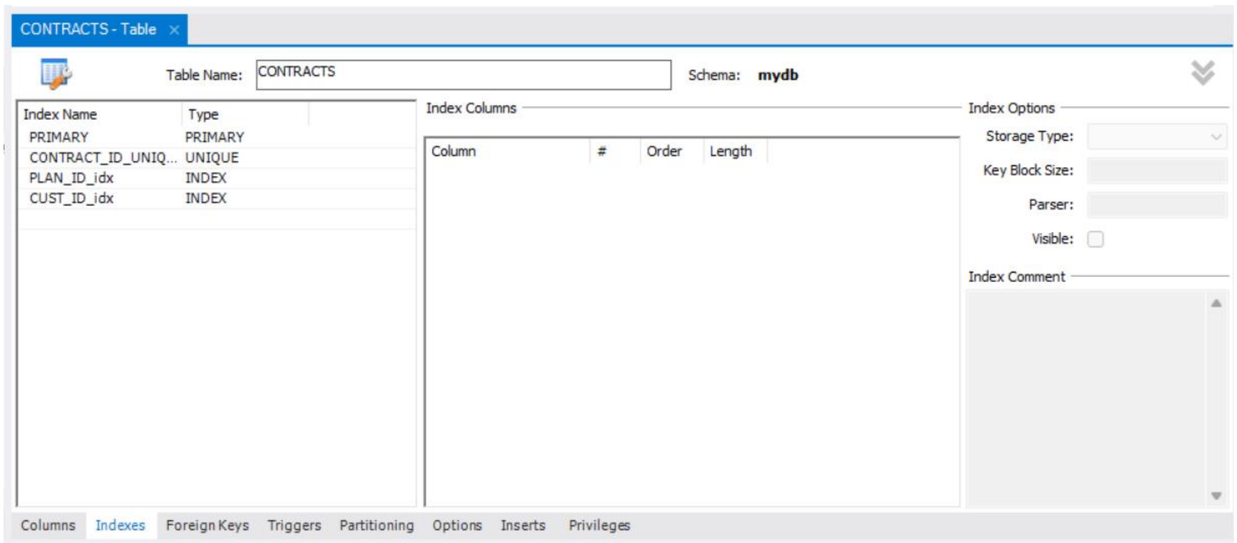

CONTRACTS - Table

Table Name: CONTRACTS Schema: **mydb**

Index Name	Type
PRIMARY	PRIMARY
CONTRACT_ID_UNIQ...	UNIQUE
PLAN_ID_idx	INDEX
CUST_ID_idx	INDEX

Index Columns

Column	#	Order	Length

Index Options
Storage Type:
Key Block Size:
Parser:
Visible: ☐

Index Comment

Columns Indexes Foreign Keys Triggers Partitioning Options Inserts Privileges

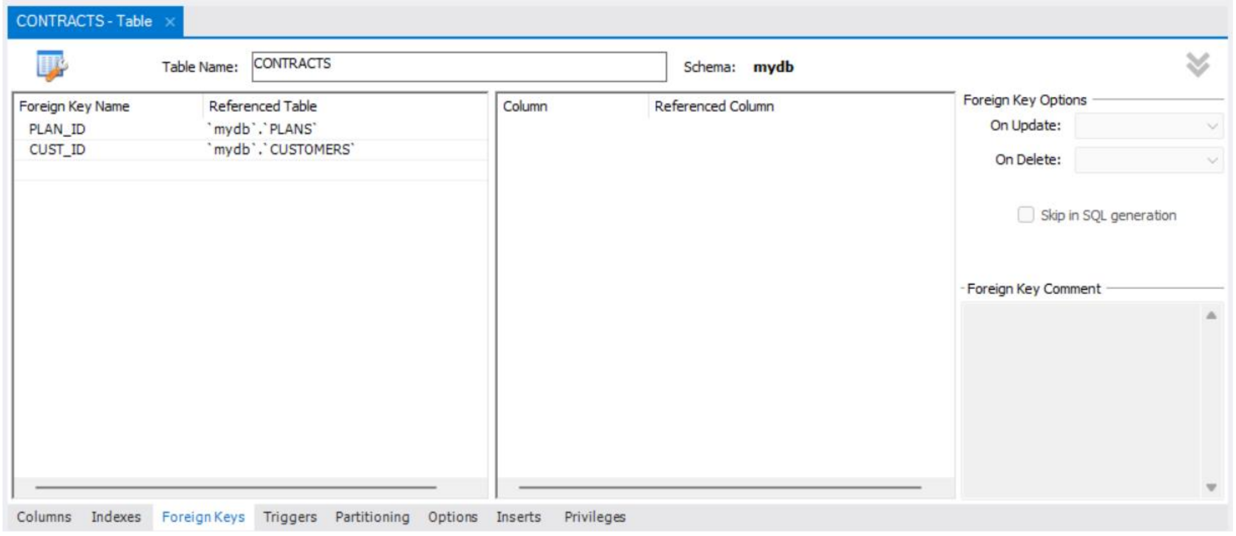

CONTRACTS - Table

Table Name: CONTRACTS Schema: **mydb**

Foreign Key Name	Referenced Table
PLAN_ID	\`mydb\`.\`PLANS\`
CUST_ID	\`mydb\`.\`CUSTOMERS\`

Column	Referenced Column

Foreign Key Options
On Update:
On Delete:
☐ Skip in SQL generation

Foreign Key Comment

Columns Indexes Foreign Keys Triggers Partitioning Options Inserts Privileges

- Table CALLS:
  CALL_ID is the primary key of the table and CONTRACT_ID is a foreign key from table CONTRACTS.

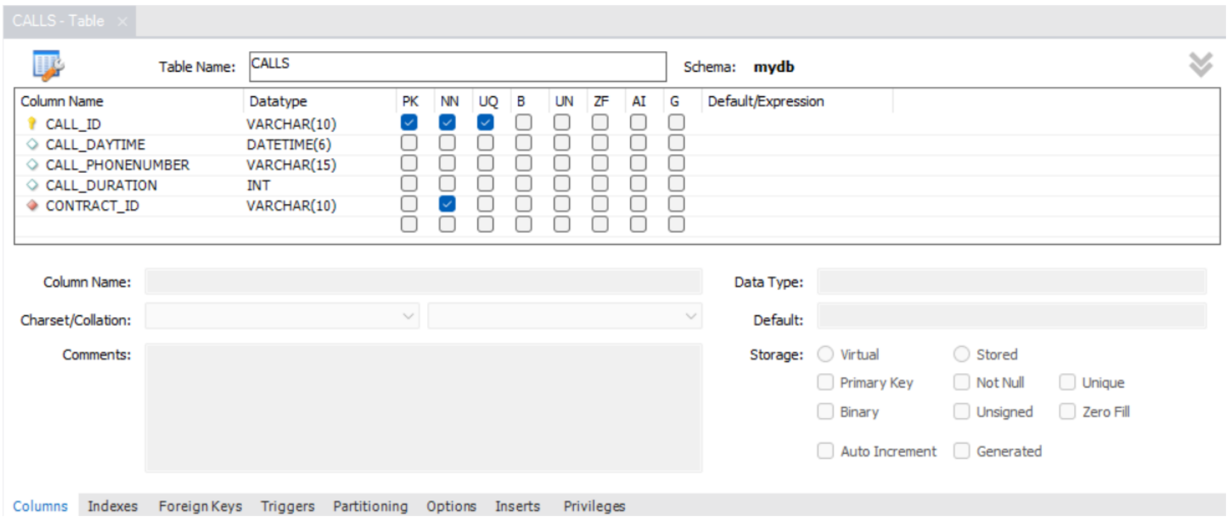

CALLS - Table

Table Name: CALLS Schema: **mydb**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
CALL_ID	VARCHAR(10)	✓	✓	✓	☐	☐	☐	☐	☐	
CALL_DAYTIME	DATETIME(6)	☐	☐	☐	☐	☐	☐	☐	☐	
CALL_PHONENUMBER	VARCHAR(15)	☐	☐	☐	☐	☐	☐	☐	☐	
CALL_DURATION	INT	☐	☐	☐	☐	☐	☐	☐	☐	
CONTRACT_ID	VARCHAR(10)	☐	✓	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Column Name:
Charset/Collation:
Comments:
Data Type:
Default:
Storage: ○ Virtual ○ Stored
☐ Primary Key ☐ Not Null ☐ Unique
☐ Binary ☐ Unsigned ☐ Zero Fill
☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options Inserts Privileges

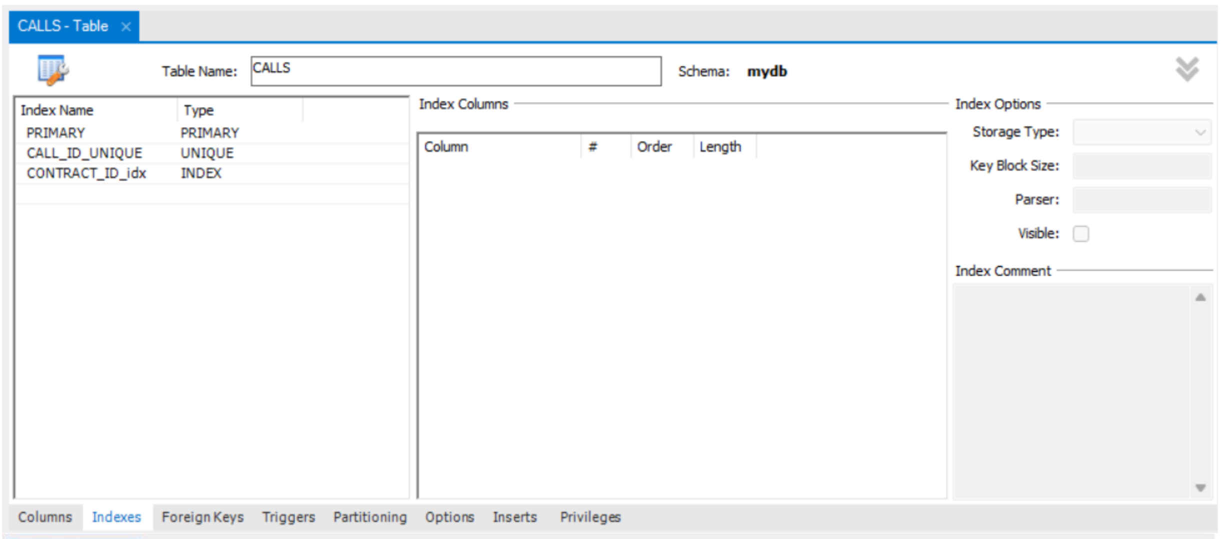

CALLS - Table

Table Name: CALLS Schema: **mydb**

Index Name	Type
PRIMARY	PRIMARY
CALL_ID_UNIQUE	UNIQUE
CONTRACT_ID_idx	INDEX

Index Columns

Column	#	Order	Length

Index Options
Storage Type:
Key Block Size:
Parser:
Visible: ☐

Index Comment

Columns | Indexes | Foreign Keys | Triggers | Partitioning | Options | Inserts | Privileges

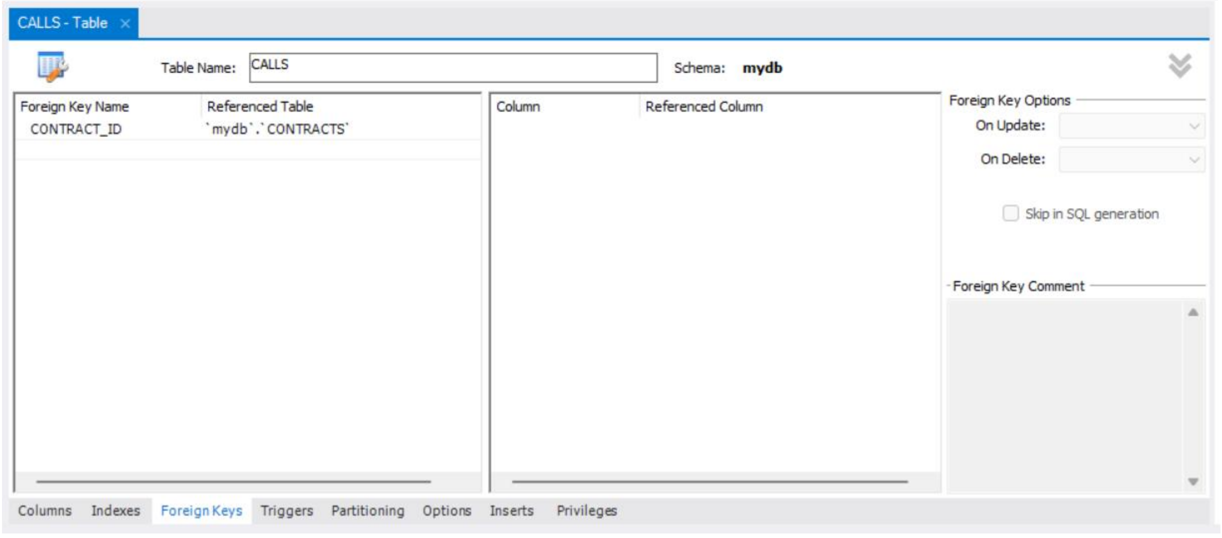

CALLS - Table

Table Name: CALLS Schema: **mydb**

Foreign Key Name	Referenced Table
CONTRACT_ID	\`mydb\`.\`CONTRACTS\`

Column	Referenced Column

Foreign Key Options
On Update:
On Delete:
☐ Skip in SQL generation

Foreign Key Comment

Columns | Indexes | Foreign Keys | Triggers | Partitioning | Options | Inserts | Privileges

- Table PLANS:
  PLAN_ID is the primary key of the table.

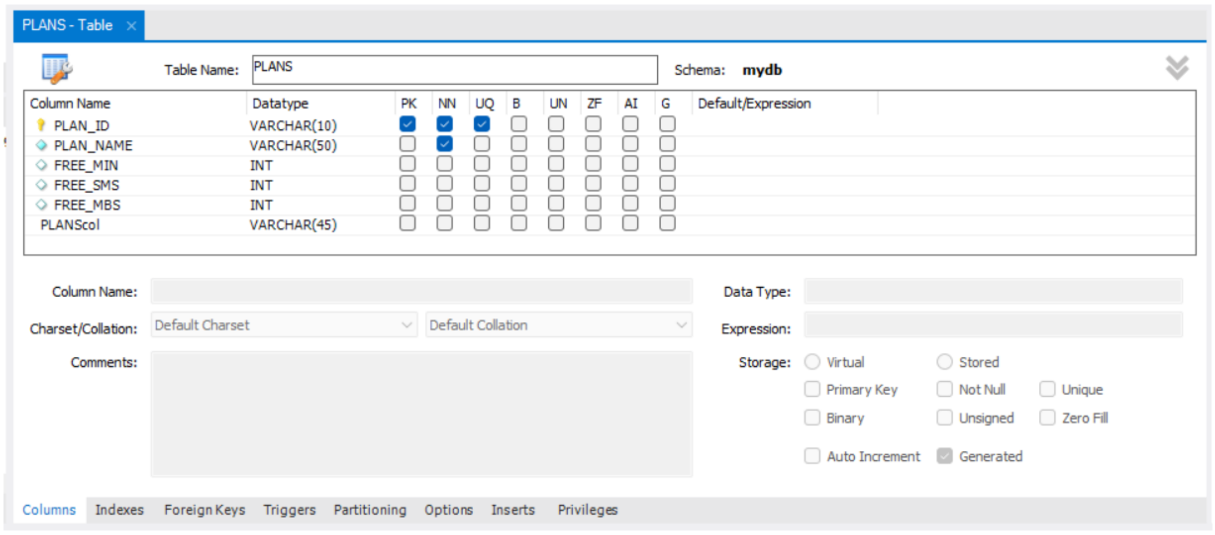

PLANS - Table

Table Name: PLANS Schema: **mydb**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
PLAN_ID	VARCHAR(10)	✓	✓	✓	☐	☐	☐	☐	☐	
PLAN_NAME	VARCHAR(50)	☐	✓	☐	☐	☐	☐	☐	☐	
FREE_MIN	INT	☐	☐	☐	☐	☐	☐	☐	☐	
FREE_SMS	INT	☐	☐	☐	☐	☐	☐	☐	☐	
FREE_MBS	INT	☐	☐	☐	☐	☐	☐	☐	☐	
PLANScol	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	

Column Name:
Charset/Collation: Default Charset Default Collation
Comments:

Data Type:
Expression:
Storage: ○ Virtual ○ Stored
☐ Primary Key ☐ Not Null ☐ Unique
☐ Binary ☐ Unsigned ☐ Zero Fill
☐ Auto Increment ☐ Generated

Columns | Indexes | Foreign Keys | Triggers | Partitioning | Options | Inserts | Privileges

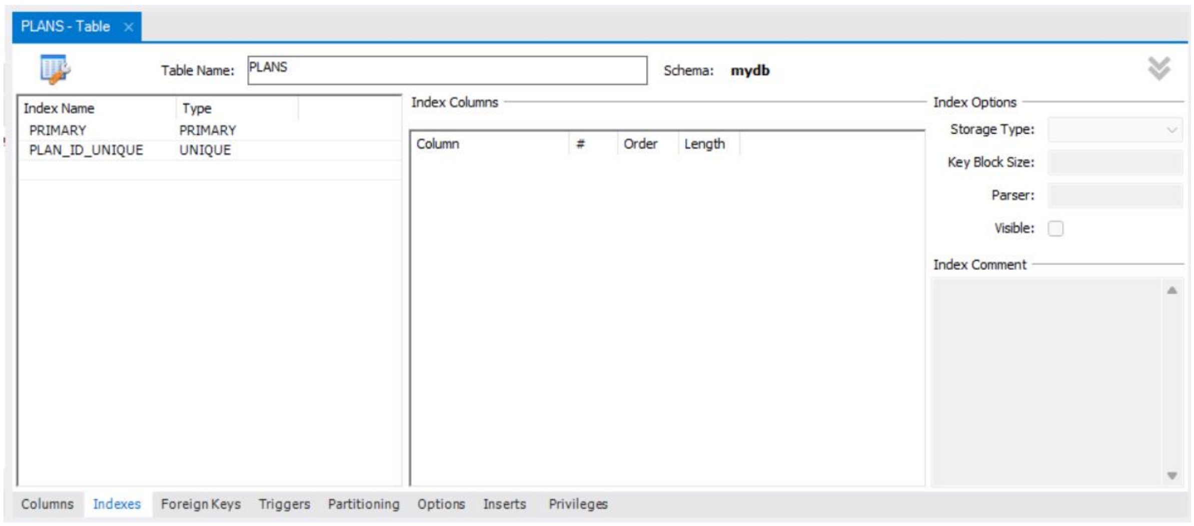


Table Name: PLANS Schema: mydb

Index Name	Type
PRIMARY	PRIMARY
PLAN_ID_UNIQUE	UNIQUE

Index Columns

Column	#	Order	Length

Index Options
Storage Type:
Key Block Size:
Parser:
Visible:
Index Comment

<u>Description of interaction between tables:</u>

- Each customer can live in only one city, therefore the **CITY** (has been indexed) is used as a **foreign key** "CITY" in the **CUSTOMERS** table. The (indexed) CITY is able to be used as **a foreign key** for multiple customers.
- Each contract can only be held by one customer, therefore the **CUSTOMER_ID**, **primary key** in **CUSTOMERS** table is used as a **foreign key** "CUSTOMER_ID" in the **CONTRACTS** table. This **CUSTOMER_ID** is able to be used as a **foreign key** for multiple contracts, as a person can have more than one contract with the company.
- Each contract can be associated with only one plan, therefore, the **PLAN_ID**, **primary key** in **PLANS** table is used as a **foreign key** "PLAN_ID" in the **CONTRACTS** table. This **PLAN_ID** is able to be used as a **foreign key** for multiple contracts.
- Each call can be made from only one phone number and therefore from only one contract. So, the **CONTRACT_ID**, **primary key** in **CONTRACTS** table, is used as a **foreign key** in **CALLS** table. This **CONTRACT_ID** is able to be used as a **foreign key** for multiple calls, since many calls can be executed by a specific phone number.

Creation of tables

After the creation of the required tables, the next step is to create some values which will be entered into them and will be used to perform the upcoming queries.

<u>Below is the code for the tables' creation.</u>

```
CREATE TABLE CITIES (
CITY_ID VARCHAR(50) NOT NULL,
CITY VARCHAR(50) NOT NULL,
POPULATION INT,
INCOME DECIMAL(10,2),

PRIMARY KEY (CITY_ID),
INDEX idx_city (CITY)  -- This line is added to create an index on CITY column
);
```

```
CREATE TABLE CUSTOMERS (
CUST_ID VARCHAR(10) NOT NULL,
FIRST_NAME VARCHAR(50) NOT NULL,
LAST_NAME VARCHAR(50) NOT NULL,
BIRTH_DATE DATE,
GENDER ENUM('Male', 'Female') NOT NULL,
CITY VARCHAR(50) NOT NULL,

 PRIMARY KEY (CUST_ID),
 FOREIGN KEY (CITY) REFERENCES CITIES(CITY)
);

CREATE TABLE PLANS (
PLAN_ID VARCHAR(10) NOT NULL,
PLAN_NAME VARCHAR(50) NOT NULL,
FREE_MIN INT,
FREE_SMS INT,
FREE_MBS INT,

PRIMARY KEY (PLAN_ID)
);

CREATE TABLE CONTRACTS (
CONTRACT_ID VARCHAR(10) NOT NULL,
PHONE_NUMBER VARCHAR(15),
START_DATE DATE,
END_DATE DATE,
CONTR_DESCRIPTION VARCHAR(255),
PLAN_ID VARCHAR(10) NOT NULL,
CUST_ID VARCHAR(10) NOT NULL,

PRIMARY KEY (CONTRACT_ID),

FOREIGN KEY (PLAN_ID) REFERENCES PLANS(PLAN_ID),
FOREIGN KEY (CUST_ID) REFERENCES CUSTOMERS(CUST_ID)
);

CREATE TABLE CALLS (
CALL_ID VARCHAR(10) NOT NULL,
CALL_DAYTIME DATETIME(6),
CALL_PHONENUMBER VARCHAR(15),
CALL_DURATION INT,
CONTRACT_ID VARCHAR(10) NOT NULL,

PRIMARY KEY (CALL_ID),
FOREIGN KEY (CONTRACT_ID) references CONTRACTS(CONTRACT_ID)
);
```

Creation of records

Now follows the insertion of the variables:

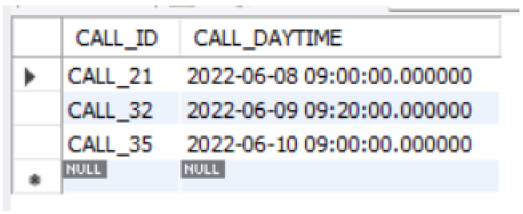
```
INSERT INTO CITIES
VALUES
('US', 'New York City', 8400000, 127100),
('FR', 'Paris', 2200000, 54100),
('ES', 'Barcelona', 1600000, 27000),
('IT', 'Rome', 2800000, 35005),
('AU', 'Sydney', 5400000, 80000);

INSERT INTO CUSTOMERS
VALUES
('CUST_01', 'Sven', 'Anders', '2000-08-16', 'Male', 'New York City'),
('CUST_02', 'Antonio', 'Moreno', '1960-02-25', 'Male', 'Barcelona'),
('CUST_03', 'Thomas', 'Hardy', '1960-02-25', 'Male', 'Paris'),
('CUST_04', 'Mary', 'Carnes', '1994-02-25', 'Female', 'Sydney'),
('CUST_05', 'Rosmary', 'Lincoln', '1988-02-25', 'Female', 'Rome'),
('CUST_06', 'Bob', 'Ashworth', '1969-02-25', 'Male', 'Rome'),
('CUST_07', 'Kathy', 'Simpson', '1962-02-25', 'Female', 'New York City'),
('CUST_08', 'Stuart', 'Chang', '1999-02-25', 'Male', 'Barcelona'),
('CUST_09', 'Helen', 'Devon', '1956-02-25', 'Female', 'Paris'),
('CUST_10', 'John', 'Afonso', '1974-02-25', 'Female', 'Sydney'),
('CUST_11', 'Paul', 'Cruz', '1975-02-25', 'Male', 'Sydney'),
('CUST_12', 'Laurie', 'Franken', '1982-02-25', 'Female', 'Rome'),
('CUST_13', 'Kate', 'Larsson', '1983-02-25', 'Female', 'New York City'),
('CUST_14', 'Karen', 'Rodriguez', '1991-02-25', 'Female', 'Barcelona'),
('CUST_15', 'Peter', 'Steel', '2001-02-25', 'Male', 'Barcelona'),
('CUST_16', 'Mario', 'Porto', '1974-03-30', 'Male', 'Rome'),
('CUST_17', 'Luigi', 'Keil', '1962-12-18', 'Male', 'Rome'),
('CUST_18', 'Kathy', 'Baley', '1964-02-20', 'Female', 'Paris');

INSERT INTO PLANS
VALUES
('P_01', 'Freedom Basic', 200, 2000, 1600),
('P_02', 'Talk_to_all', 300, 2500, 2000),
('P_03', 'Exclusive', 400, 3000, 2400),
('P_04', 'Freedom Plus', 500, 3500, 2800),
('P_05', 'Unlimited 01', 600, 4000, 3200),
('P_06', 'Freedom Basic', 700, 4500, 3600),
('P_07', 'Exclusive', 800, 5000, 4000),
('P_08', 'Unlimited 02', 900, 5500, 4400),
('P_09', 'Freedom Plus', 1000, 6000, 4800),
('P_10', 'Freedom Basic', 1100, 6500, 5200),
('P_11', 'Talk_to_all', 800, 6500, 5200),
('P_12', 'Unlimited 01', 900, 7000, 5600),
('P_13', 'Talk_to_all', 1000, 7500, 6000),
('P_14', 'Freedom Plus', 1100, 8000, 6400),
('P_15', 'Exclusive', 1200, 8500, 6800),
('P_16', 'Talk_to_all', 800, 7000, 6800),
('P_17', 'Freedom Basic', 1000, 8500, 5200),
('P_18', 'Unlimited 02', 800, 6500, 4800);

INSERT INTO CONTRACTS
VALUES
('CON_01', '123-456-7890', '2022-01-15', '2023-01-14', 'Standard Plan', 'P_01','CUST_01'),
('CON_02', '987-654-3210', '2022-03-20', '2023-03-19', 'Premium Plan', 'P_02', 'CUST_02'),
('CON_03', '555-123-4567', '2022-04-10', '2023-04-09', 'Basic Plan', 'P_03', 'CUST_03'),
('CON_04', '111-222-3333', '2022-02-05', '2023-02-04', 'Business Plan', 'P_04', 'CUST_04'),
('CON_05', '444-555-6666', '2022-05-01', '2023-04-30', 'Standard Plan', 'P_05', 'CUST_05'),
('CON_06', '888-777-9999', '2022-06-10', '2023-06-09', 'Premium Plan', 'P_06', 'CUST_06'),
('CON_07', '666-777-8888', '2022-08-15', '2023-08-14', 'Basic Plan', 'P_07', 'CUST_07'),
('CON_08', '333-444-5555', '2022-09-20', '2023-09-19', 'Business Plan', 'P_08', 'CUST_08'),
('CON_09', '777-888-9999', '2022-07-05', '2023-07-04', 'Standard Plan', 'P_09', 'CUST_09'),
('CON_10', '222-333-4444', '2022-10-01', '2023-09-30', 'Premium Plan', 'P_10', 'CUST_10'),
('CON_11', '555-777-9999', '2022-04-01', '2023-03-31', 'Business Plan', 'P_11', 'CUST_11'),
('CON_12', '111-333-5555', '2022-03-15', '2023-03-14', 'Standard Plan', 'P_12', 'CUST_12'),
('CON_13', '333-555-7777', '2022-02-01', '2023-01-31', 'Premium Plan', 'P_13', 'CUST_13'),
('CON_14', '999-777-5555', '2022-01-20', '2023-01-19', 'Basic Plan', 'P_14', 'CUST_14'),
('CON_15', '222-444-6666', '2022-06-10', '2023-06-09', 'Business Plan', 'P_15', 'CUST_15'),
('CON_16', '222-333-5555', '2022-09-20', '2023-03-17', 'Standard Plan', 'P_16', 'CUST_16'),
('CON_17', '333-444-9999', '2022-12-18', '2023-05-21', 'Premium Plan', 'P_17', 'CUST_17'),
('CON_18', '888-222-7890', '2022-01-11', '2023-07-09', 'Business Plan', 'P_18', 'CUST_18');

INSERT INTO CALLS
VALUES
('CALL_01', '2020-01-10 08:30:00', '123-456-7890', 180, 'CON_05'),
('CALL_02', '2020-02-20 10:45:00', '987-654-3210', 210, 'CON_04'),
('CALL_03', '2021-01-10 08:30:00', '555-123-4567', 90, 'CON_03'),
('CALL_04', '2021-01-01 10:00:00', '555-666-7777', 100, 'CON_02'),
('CALL_05', '2022-01-05 16:20:00', '444-555-6666', 240, 'CON_01'),
('CALL_06', '2022-02-15 09:15:00', '123-456-7890', 25, 'CON_10'),
('CALL_07', '2023-01-20 09:30:00', '123-456-7890', 20, 'CON_09'),
('CALL_08', '2023-02-01 13:45:00', '333-444-5555', 90, 'CON_08'),
('CALL_09', '2020-03-10 14:30:00', '555-666-7777', 150, 'CON_07'),
('CALL_10', '2020-04-15 16:45:00', '777-888-9999', 180, 'CON_06'),
('CALL_11', '2021-03-05 11:20:00', '222-111-0000', 120, 'CON_17'),
('CALL_12', '2021-04-10 10:30:00', '666-777-8888', 210, 'CON_16'),
('CALL_13', '2022-03-15 14:15:00', '888-999-1111', 190, 'CON_15'),
('CALL_14', '2022-04-20 16:30:00', '111-222-3333', 160, 'CON_14'),
('CALL_15', '2023-03-25 09:45:00', '999-888-7777', 220, 'CON_13'),
('CALL_16', '2023-04-10 11:00:00', '555-444-3333', 130, 'CON_12'),
('CALL_17', '2023-05-08 11:59:00', '111-222-3333', 123, 'CON_01'),
('CALL_18', '2022-11-01 13:45:00', '555-123-4567', 90, 'CON_02'),
('CALL_19', '2022-12-15 10:00:00', '111-222-3333', 120, 'CON_03'),
('CALL_20', '2021-02-01 11:00:00', '444-555-6666', 120, 'CON_04'),
('CALL_21', '2022-06-08 09:00:00', '123-456-7890', 15, 'CON_05'),
('CALL_22', '2023-04-01 12:15:00', '333-444-5555', 90, 'CON_06'),
('CALL_23', '2022-09-01 15:30:00', '333-222-1111', 200, 'CON_07'),
('CALL_24', '2023-05-15 09:30:00', '777-888-9999', 120, 'CON_08'),
('CALL_25', '2023-06-01 16:20:00', '222-333-4444', 240, 'CON_09'),
('CALL_26', '2021-07-10 14:00:00', '555-777-9999', 180, 'CON_10'),
('CALL_27', '2022-08-05 11:15:00', '111-333-5555', 210, 'CON_11'),
('CALL_28', '2021-10-01 16:20:00', '444-777-8888', 240, 'CON_12'),
('CALL_29', '2022-10-15 10:00:00', '999-777-5555', 120, 'CON_13'),
('CALL_30', '2022-10-01 17:15:00', '444-777-8888', 210, 'CON_14'),
('CALL_31', '2022-12-10 08:30:00', '123-456-7890', 180, 'CON_15'),
('CALL_32', '2022-06-09 09:20:00', '123-456-7890', 29, 'CON_16'),
('CALL_33', '2023-02-01 12:15:00', '555-123-4567', 90, 'CON_17'),
('CALL_34', '2021-03-15 11:30:00', '111-222-3333', 120, 'CON_03'),
('CALL_35', '2022-06-10 09:00:00', '123-456-7890', 28, 'CON_14'),
('CALL_36', '2022-07-21 08:15:00', '222-222-9999', 170, 'CON_18');
```

DELIVERABLE 3

Queries – SQL coding

a) *Show the call id of all calls that were made between 8am and 10am in June 2022 having duration < 30.*

```
SELECT CALL_ID, CALL_DAYTIME
FROM CALLS
WHERE
    EXTRACT(YEAR_MONTH FROM CALL_DAYTIME) = 202206
    AND EXTRACT(HOUR_MINUTE FROM CALL_DAYTIME) BETWEEN 800 AND 1000
    AND CALL_DURATION < 30;
```



CALL_ID	CALL_DAYTIME
CALL_21	2022-06-08 09:00:00.000000
CALL_32	2022-06-09 09:20:00.000000
CALL_35	2022-06-10 09:00:00.000000
NULL	NULL

This SQL query selects **CALL_ID** and **CALL_DAYTIME** from the **CALLS** table with specific conditions. It retrieves call records that meet the following criteria:

- The **CALL_DAYTIME** is in the year and month June 2022 (202206)

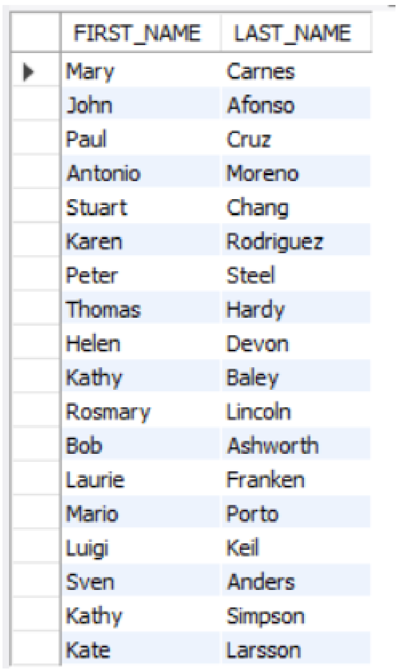
- The time of day (hour and minute) extracted from **CALL_DAYTIME** falls between 8:00 AM (800) and 10:00 AM (1000)
- The **CALL_DURATION** is less than 30 minutes

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

```
SELECT FIRST_NAME, LAST_NAME
from CUSTOMERS
WHERE CITY IN (
   SELECT CITY
   FROM CITIES
   WHERE POPULATION > 20000
);
```

This code retrieves the first and last names of customers from the **CUSTOMERS** table who reside in cities with a population exceeding 20.000, using a subquery to identify the eligible cities based on population criteria.

- The main query selects the **FIRST_NAME** and **LAST_NAME** columns from the **CUSTOMERS** table
- The **WHERE** clause filters the results, using a subquery to determine which cities have a population greater than 20.000
- The subquery selects the **CITY** column from the **CITIES** table
- The main query checks if the **CITY** of each customer in the **CUSTOMERS** table is in the list of cities obtained from the subquery. In other words, it filters for customers who live in cities with a population greater than 20.000

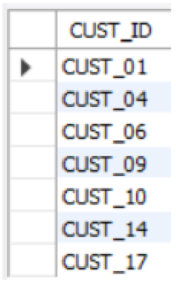


FIRST_NAME	LAST_NAME
Mary	Carnes
John	Afonso
Paul	Cruz
Antonio	Moreno
Stuart	Chang
Karen	Rodriguez
Peter	Steel
Thomas	Hardy
Helen	Devon
Kathy	Baley
Rosmary	Lincoln
Bob	Ashworth
Laurie	Franken
Mario	Porto
Luigi	Keil
Sven	Anders
Kathy	Simpson
Kate	Larsson

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

```
SELECT CUST_ID
FROM CONTRACTS
WHERE PLAN_ID IN (
   SELECT PLAN_ID
   FROM PLANS
   WHERE PLAN_NAME LIKE 'Freedom%'
);
```

This SQL query retrieves customer IDs for contracts associated with plans whose names start with 'Freedom%' by using a subquery to find the relevant **PLAN_ID** values in the **PLANS** table and then checking for matches in the **CONTRACTS** table.



CUST_ID
CUST_01
CUST_04
CUST_06
CUST_09
CUST_10
CUST_14
CUST_17

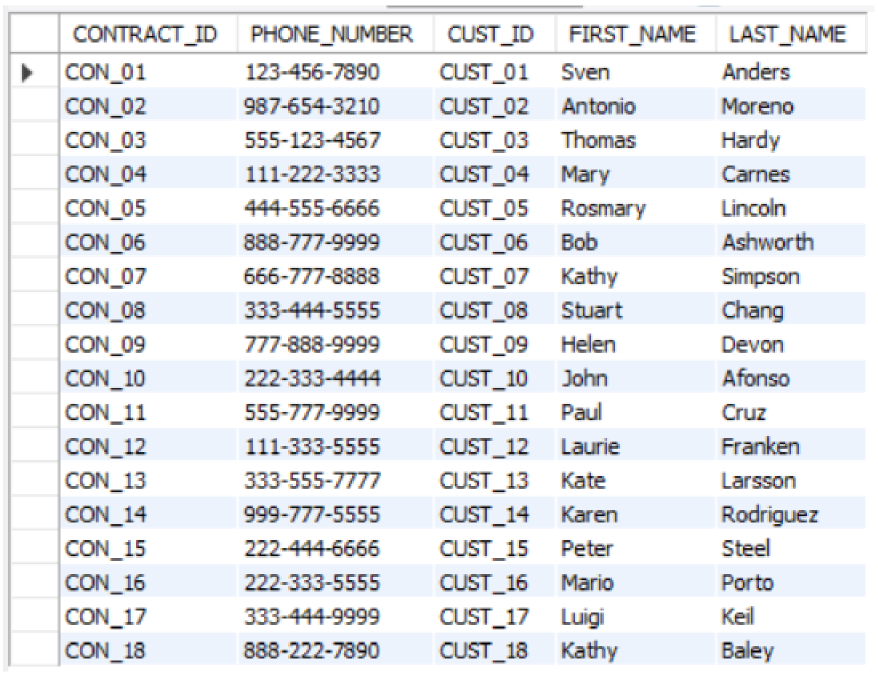
- The subquery in the **WHERE** clause:
    i. It selects **PLAN_ID** values from the **PLANS** table

ii. It filters the results to find **PLAN_NAME** values that start with 'Freedom' using the **LIKE** operator and a wildcard '%'

- The main query:
  i. It selects **CUST_ID** values from the **CONTRACTS** table
  ii. The **WHERE** clause filters the results to include only rows where the **PLAN_ID** matches any of the **PLAN_ID** values obtained from the subquery

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 CNTR.CONTRACT_ID, CNTR.PHONE_NUMBER, CNTR.CUST_ID, CST.FIRST_NAME,
CST.LAST_NAME
FROM CONTRACTS AS CNTR
JOIN CUSTOMERS AS CST ON CNTR.CUST_ID = CST.CUST_ID
WHERE CNTR.END_DATE <= DATE_ADD(current_date(), interval 60 DAY);
```



CONTRACT_ID	PHONE_NUMBER	CUST_ID	FIRST_NAME	LAST_NAME
CON_01	123-456-7890	CUST_01	Sven	Anders
CON_02	987-654-3210	CUST_02	Antonio	Moreno
CON_03	555-123-4567	CUST_03	Thomas	Hardy
CON_04	111-222-3333	CUST_04	Mary	Carnes
CON_05	444-555-6666	CUST_05	Rosmary	Lincoln
CON_06	888-777-9999	CUST_06	Bob	Ashworth
CON_07	666-777-8888	CUST_07	Kathy	Simpson
CON_08	333-444-5555	CUST_08	Stuart	Chang
CON_09	777-888-9999	CUST_09	Helen	Devon
CON_10	222-333-4444	CUST_10	John	Afonso
CON_11	555-777-9999	CUST_11	Paul	Cruz
CON_12	111-333-5555	CUST_12	Laurie	Franken
CON_13	333-555-7777	CUST_13	Kate	Larsson
CON_14	999-777-5555	CUST_14	Karen	Rodriguez
CON_15	222-444-6666	CUST_15	Peter	Steel
CON_16	222-333-5555	CUST_16	Mario	Porto
CON_17	333-444-9999	CUST_17	Luigi	Keil
CON_18	888-222-7890	CUST_18	Kathy	Baley

This retrieves contract and customer information for contracts that are set to end within the next 60 days, providing details about the contract, phone number, customer ID, and customer's first and last names.

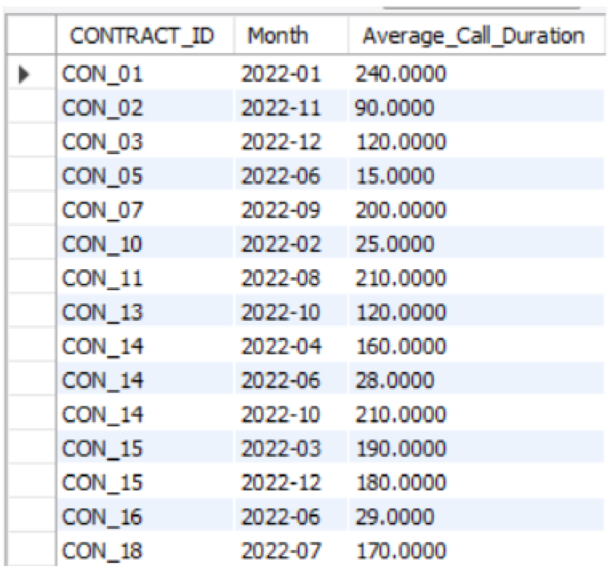
- The **SELECT** clause selects several columns:
  i. **CONTRACT_ID**, **PHONE_NUMBER**, and **CUST_ID** from the **CONTRACTS** table, aliased as **CNTR**
  ii. **FIRST_NAME** and **LAST_NAME** from the **CUSTOMERS** table, aliased as **CST**
- The **FROM** clause specifies that the data comes from the **CONTRACTS** table (aliased as **CNTR**) and performs an **INNER JOIN** operation with the **CUSTOMERS** table (aliased as **CST) based on**

the common column CUST_ID`. This join links contract information to the associated customer information

- The **WHERE** clause filters the results based on a condition:
    i. It checks if the **END_DATE** of each contract (**CNTR.END_DATE**) is less than or equal to the date 60 days from the current date (using **DATE_ADD(current_date(), INTERVAL 60 DAY)**). This condition retrieves contracts that are due to end within the next 60 days

e) *For each contract id and each month of 2022, show the average duration of calls*

```
SELECT
  C.CONTRACT_ID,
  DATE_FORMAT(C.CALL_DAYTIME, '%Y-%m') AS Month,
  AVG(C.CALL_DURATION) AS Average_Call_Duration
FROM CALLS AS C
WHERE YEAR(C.CALL_DAYTIME) = 2022
GROUP BY C.CONTRACT_ID, Month
ORDER BY C.CONTRACT_ID, Month;
```



CONTRACT_ID	Month	Average_Call_Duration
CON_01	2022-01	240.0000
CON_02	2022-11	90.0000
CON_03	2022-12	120.0000
CON_05	2022-06	15.0000
CON_07	2022-09	200.0000
CON_10	2022-02	25.0000
CON_11	2022-08	210.0000
CON_13	2022-10	120.0000
CON_14	2022-04	160.0000
CON_14	2022-06	28.0000
CON_14	2022-10	210.0000
CON_15	2022-03	190.0000
CON_15	2022-12	180.0000
CON_16	2022-06	29.0000
CON_18	2022-07	170.0000

This SQL query calculates and presents the average call duration for each combination of contract and month in the year 2022.

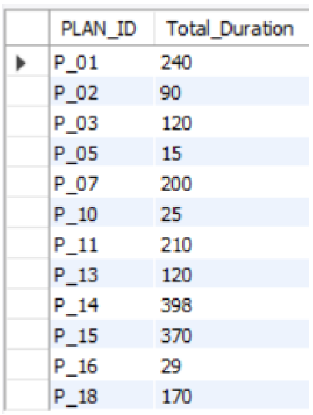
- The **SELECT** clause selects three columns:
    i. **CONTRACT_ID** from the **CALLS** table (aliased as C)
    ii. The formatted month (year and month) of the call time, using the **DATE_FORMAT** function and aliased as **Month**
    iii. The average call duration, calculated with the **AVG** function and aliased as **Average_Call_Duration**
- The **FROM** clause specifies that the data comes from the **CALLS** table, and it is given the alias **c**
- The **WHERE** clause filters the results to include only records where the year of the **CALL_DAYTIME** is equal to 2022

- The **GROUP BY** clause groups the results by **CONTRACT_ID** and **Month**. It calculates the average call duration for each unique combination of contract and month
- The **ORDER BY** clause orders the results first by **CONTRACT_ID** and then by **Month**

f) *Show the total duration of calls in 2022 per plan id*

```
SELECT CNTR.PLAN_ID, SUM(C.CALL_DURATION) AS Total_Duration
FROM CALLS AS C
INNER JOIN CONTRACTS AS CNTR ON C.CONTRACT_ID = CNTR.CONTRACT_ID
WHERE YEAR(C.CALL_DAYTIME) = 2022
GROUP BY CNTR.PLAN_ID;
```

This SQL query retrieves data from the **CALLS** (aliased as C) and **CONTRACTS** (aliased as CNTR) tables and calculates the total call duration for each unique **PLAN_ID** associated with contracts in the year 2022.



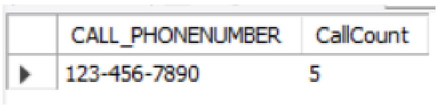
PLAN_ID	Total_Duration
P_01	240
P_02	90
P_03	120
P_05	15
P_07	200
P_10	25
P_11	210
P_13	120
P_14	398
P_15	370
P_16	29
P_18	170

- The **SELECT** clause selects the following columns:
  i. **PLAN_ID** from the **CONTRACTS** table
  ii. The total call duration for each **PLAN_ID** calculated using **SUM(C.CALL_DURATION)**, aliased as **Total_Duration**
- The **FROM** clause specifies that the data is coming from the **CALLS** table and the **CONTRACTS** table, and it performs an **INNER JOIN** operation between them using the common column **CONTRACT_ID**
- The **WHERE** clause filters the results to include only records where the year of the **CALL_DAYTIME** is equal to 2022
- The **GROUP BY** clause groups the results by **PLAN_ID**, calculating the total call duration for each unique **PLAN_ID**

g) *Show the top called number among TP's customers in 2022*

```
SELECT CALL_PHONENUMBER, COUNT(*) AS CallCount
FROM CALLS
WHERE YEAR(CALL_DAYTIME) = 2022
GROUP BY CALL_PHONENUMBER
ORDER BY CallCount DESC
LIMIT 1;
```

This SQL query retrieves the phone number (**CALL_PHONENUMBER**) with the highest call count in the year 2022.



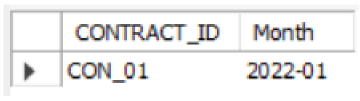
CALL_PHONENUMBER	CallCount
123-456-7890	5

- The **SELECT** clause selects two columns:
  i. **CALL_PHONENUMBER** from the **CALLS** table
  ii. The count of records for each phone number, aliased as **CallCount**
- The **FROM** clause specifies that the data is coming from the **CALLS** table
- The **WHERE** clause filters the results to include only records where the year of the **CALL_DAYTIME** is equal to 2022

- The **GROUP BY** clause groups the results by **CALL_PHONENUMBER**, counting the number of records for each phone number
- The **ORDER BY** clause orders the results in descending order based on **CallCount**, so the phone number with the highest call count will appear at the top
- The **LIMIT 1** clause ensures that only the top result (highest call count) is returned

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*

```
SELECT
  C.CONTRACT_ID,
  DATE_FORMAT(C.CALL_DAYTIME, '%Y-%m') AS Month
FROM CALLS AS C
INNER JOIN CONTRACTS AS CNTR ON C.CONTRACT_ID = CNTR.CONTRACT_ID
INNER JOIN PLANS AS P ON CNTR.PLAN_ID = P.PLAN_ID
WHERE YEAR(C.CALL_DAYTIME) = 2022
GROUP BY C.CONTRACT_ID, Month, P.FREE_MIN
HAVING SUM(C.CALL_DURATION) > P.FREE_MIN;
```



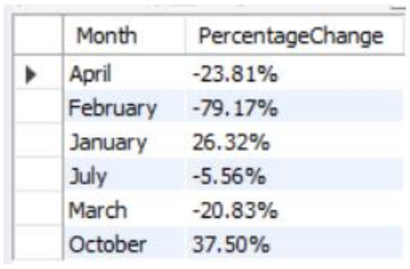
	CONTRACT_ID	Month
▶	CON_01	2022-01

This SQL query retrieves data related to contracts, their associated call records, and plans. It filters and groups the results to find contracts that exceeded their free minutes limit in the year 2022.

- The **SELECT** clause selects two columns:
  1. **CONTRACT_ID** from the **CALLS** table (aliased as **C**)
  2. The formatted month (year and month) of the call time from the **CALLS** table (aliased as **Month**)
- The **FROM** clause specifies that the data comes from the **CALLS** table (aliased as **C**) and performs an **INNER JOIN** operation with the **CONTRACTS** table (aliased as **CNTR**) based on the **CONTRACT_ID**. It then joins with the **PLANS** table (aliased as **P**) based on the **PLAN_ID**
- The **WHERE** clause filters the results to include only records where the year of the call time is equal to 2022
- The **GROUP BY** clause groups the results by **CONTRACT_ID**, **Month** (formatted call time), and **P.FREE_MIN** (the free minutes associated with the plan)
- The **HAVING** clause filters the grouped results to include only those where the sum of call durations (**SUM(C.CALL_DURATION)**) is greater than the free minutes (**P.FREE_MIN**) associated with the plan. In other words, it selects contracts where the total call duration in a month exceeded the allocated free minutes

i) *For each month of 2022, show the percentage change of the total duration of calls compared to the same month of 2021*

```
SELECT
  MONTHNAME(CALL_DAYTIME) AS Month,
    CONCAT(
        FORMAT(
            ((SUM(CASE WHEN YEAR(CALL_DAYTIME) = 2022 THEN CALL_DURATION ELSE 0 END) -
            SUM(CASE WHEN YEAR(CALL_DAYTIME) = 2021 THEN CALL_DURATION ELSE 0 END)) /
            SUM(CASE WHEN YEAR(CALL_DAYTIME) = 2021 THEN CALL_DURATION ELSE 0 END) *
100),
            2
        ),
        '%'
    ) AS PercentageChange
FROM CALLS
WHERE YEAR(CALL_DAYTIME) IN (2021, 2022)
GROUP BY MONTH
HAVING SUM(CASE WHEN YEAR(CALL_DAYTIME) = 2021 THEN CALL_DURATION ELSE 0 END)
> 0
ORDER BY MONTH;
```



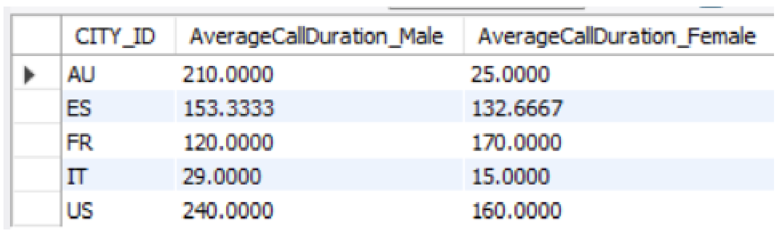
```
Month      PercentageChange
April      -23.81%
February   -79.17%
January    26.32%
July       -5.56%
March      -20.83%
October    37.50%
```

This SQL query calculates and reports the percentage change in total call duration for each month between the years 2021 and 2022. It uses conditional aggregation to calculate the change in call duration and then formats the result as a percentage change.

- The **SELECT** clause selects two columns:
  i. The name of the month (**MONTHNAME(CALL_DAYTIME)**) as "Month"
  ii. The percentage change in call duration, calculated using conditional aggregation and formatted as a percentage
- The **FROM** clause specifies that the data comes from the **CALLS** table
- The **WHERE** clause filters the results to include only records from the years 2021 and 2022
- The **GROUP BY** clause groups the results by month
- The **HAVING** clause filters the grouped results to include only those months where the total call duration in 2021 was greater than 0. This ensures that you're comparing months where there was some call activity in 2021
- The **ORDER BY** clause orders the results by month

j) *For each city id and calls made in 2022, show the average call duration by females and the average call duration by males (i.e. three columns)*

```
SELECT
    CT.CITY_ID,
   AVG(CASE WHEN CST.GENDER = 'Male' THEN C.CALL_DURATION ELSE NULL END) AS
AverageCallDuration_Male,
   AVG(CASE WHEN CST.GENDER = 'Female' THEN C.CALL_DURATION ELSE NULL END) AS
AverageCallDuration_Female
FROM CALLS AS C
INNER JOIN CONTRACTS AS CNTR ON C.CONTRACT_ID = CNTR.CONTRACT_ID
INNER JOIN CUSTOMERS AS CST ON CNTR.CUST_ID = CST.CUST_ID
INNER JOIN CITIES AS CT ON CST.CITY = CT.CITY
WHERE YEAR(C.CALL_DAYTIME) = 2022
GROUP BY CT.CITY_ID;
```



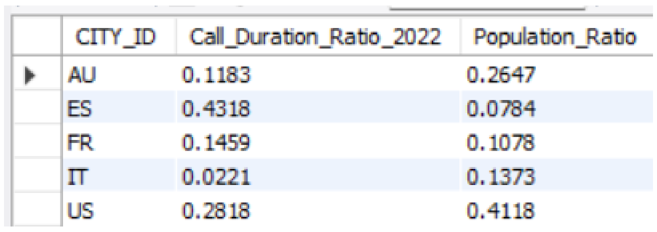
CITY_ID	AverageCallDuration_Male	AverageCallDuration_Female
AU	210.0000	25.0000
ES	153.3333	132.6667
FR	120.0000	170.0000
IT	29.0000	15.0000
US	240.0000	160.0000

This SQL query calculates and reports the average call duration for male and female customers in each city in the year 2022. It uses conditional aggregation to calculate separate averages for male and female customers.

- The **SELECT** clause selects three columns:
  i. **CT.CITY_ID** from the **CITIES** table (aliased as **CT**)
  ii. The average call duration for male customers, calculated using conditional aggregation and aliased as **AverageCallDuration_Male**
  iii. The average call duration for female customers, calculated using conditional aggregation and aliased as **AverageCallDuration_Female**
- The **FROM** clause specifies that the data comes from multiple tables: **CALLS** (aliased as **C**), **CONTRACTS** (aliased as **CNTR**), **CUSTOMERS** (aliased as **CST**), and **CITIES** (aliased as **CT**). It performs multiple **INNER JOIN** operations between these tables based on the relevant key relationships
- The **WHERE** clause filters the results to include only records from the year 2022
- The **GROUP BY** clause groups the results by **CT.CITY_ID**, which represents each city

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 2022 over the total duration of all calls made in 2022, and the ratio of the city's population over the total population of all cities (i.e three columns)

```
SELECT
   CITY.CITY_ID,
   (SELECT SUM(CALL_DURATION) FROM CALLS C2
    INNER JOIN CONTRACTS CNTR2 ON C2.CONTRACT_ID = CNTR2.CONTRACT_ID
    INNER JOIN CUSTOMERS CST2 ON CNTR2.CUST_ID = CST2.CUST_ID
    INNER JOIN CITIES CITY2 ON CST2.CITY = CITY2.CITY
    WHERE YEAR(C2.CALL_DAYTIME) = 2022 AND CITY2.CITY_ID = CITY.CITY_ID) /
   (SELECT SUM(CALL_DURATION) FROM CALLS C3 WHERE YEAR(C3.CALL_DAYTIME) = 2022)
AS 'Call_Duration_Ratio_2022',
   CITY.POPULATION /
   (SELECT SUM(POPULATION) FROM CITIES) AS 'Population_Ratio'
FROM CITIES AS CITY;
```



CITY_ID	Call_Duration_Ratio_2022	Population_Ratio
AU	0.1183	0.2647
ES	0.4318	0.0784
FR	0.1459	0.1078
IT	0.0221	0.1373
US	0.2818	0.4118

This SQL query calculates two ratios for each city. The first ratio is the "Call Duration Ratio for 2022," which compares the total call duration for the city in 2022 to the total call duration for all cities in 2022. The second ratio is the "Population Ratio," which compares the population of the city to the total population of all cities.

- The **SELECT** clause selects three columns:
  i. **CITY.CITY_ID** from the **CITIES** table (aliased as **CITY**)
  ii. The "Call Duration Ratio for 2022," calculated by dividing the total call duration for the specific city in 2022 by the total call duration for all cities in 2022. This is obtained through subqueries
  iii. The "Population Ratio," calculated by dividing the population of the specific city by the total population of all cities. This is also obtained through subqueries
- The **FROM** clause specifies that the data comes from the **CITIES** table, aliased as **CITY**

Query K – Python coding

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

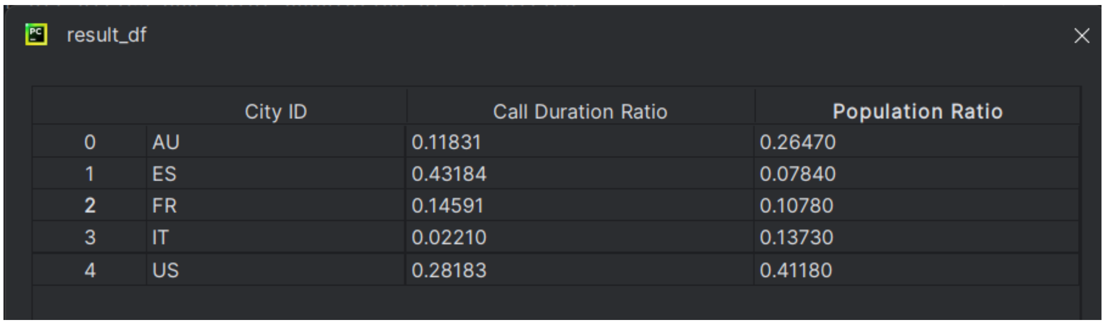
The purpose of query k was to show for each city id, the ratio of the total duration of the calls made from customers staying in that city in 2022 over the total duration of all calls made in 2022, and the ratio of the city's population over the total population of all cities.

<u>Python code:</u>

```python
import pandas as pd
import mysql.connector

# Connect to the MySQL database
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="123456789!",
    database="hrd"
)

# Define the SQL query to retrieve the required data
sql_query = """
SELECT
    CITY.CITY_ID,
    (SELECT SUM(CALL_DURATION) FROM CALLS C2
     INNER JOIN CONTRACTS CNTR2 ON C2.CONTRACT_ID = CNTR2.CONTRACT_ID
     INNER JOIN CUSTOMERS CST2 ON CNTR2.CUST_ID = CST2.CUST_ID
     INNER JOIN CITIES CITY2 ON CST2.CITY = CITY2.CITY
     WHERE YEAR(C2.CALL_DAYTIME) = 2022 AND CITY2.CITY_ID = CITY.CITY_ID) /
    (SELECT SUM(CALL_DURATION) FROM CALLS C3 WHERE YEAR(C3.CALL_DAYTIME) = 2022)
AS 'Call_Duration_Ratio_2022',
    CITY.POPULATION /
    (SELECT SUM(POPULATION) FROM CITIES) AS 'Population_Ratio'
FROM CITIES AS CITY;
"""

# Execute the query and fetch the results into a Pandas DataFrame
df = pd.read_sql(sql_query, db)

print(df)
```

Explanation of code:

- **import pandas as pd** :
  Importing library "**pandas**" aliased as "pd". This library allows us to structure data and create tables, known as "DataFrames".
- **import mysql.connector**
  db = mysql.connector.connect() :
  Importing module "**mysql.connector**". This module allows connection to data base. The link is stored in a variable named "db", so we can refer to db whenever it is necessary to use the link.
- **sql_query** :
  By this command is defined the SQL query from which we will retrieve the required data. Specifically, this query corresponds to query k from "Queries – SQL coding" section above (without performance of GROUP BY function).
- **df = pd.read_sql(sql_query, db)** :
  Executing the SQL query and fetching the results into a Pandas DataFrame.
- **print(df)** :
  Printing the results of the query into DataFrame "df".



	City ID	Call Duration Ratio	Population Ratio
0	AU	0.11831	0.26470
1	ES	0.43184	0.07840
2	FR	0.14591	0.10780
3	IT	0.02210	0.13730
4	US	0.28183	0.41180