

PROJECT PHASE II- DATABASES

TEAM MEMBERS

Hatice NUREL	180709001
Melise KAYA	180709034

(1)EXPLANATION OUR DATA

Telecom Customer Churn Prediction

The Customer Churn table contains information on all 7,043 customers from a Telecommunications company in California in Q2 2022. Each record represents one customer, and contains details about their demographics, location, tenure, subscription services, status for the quarter (joined, stayed, or churned), and more! The Zip Code Population table contains complimentary information on the estimated populations for the California zip codes in the Customer Churn table.

- How many files exist?
Three file
- How many rows and columns do they contain?
There is 38 columns and 7044 rows.
- How many string(non-numeric) columns do you have?
String :13
Boolean: 13
Integer: 9

(2)CHANGES/ADDITIONS

We used the RDF method because all our data has the same relationship type. Our entity has completely changed. We have specified null values in our data and split columns containing two data types. In this way, we created a questionable database.

SOURCES OF YOUR DATA

<https://www.kaggle.com/datasets/shilongzhuang/telecom-customer-churn-by-maven-analytics>

(3)LOADING DATA

We divide our whole data into separate tables according to our entity. We use python to parse our dataset using csv. Then we loaded the dataset part by part to model and we use forward engineering. We could upload all data.

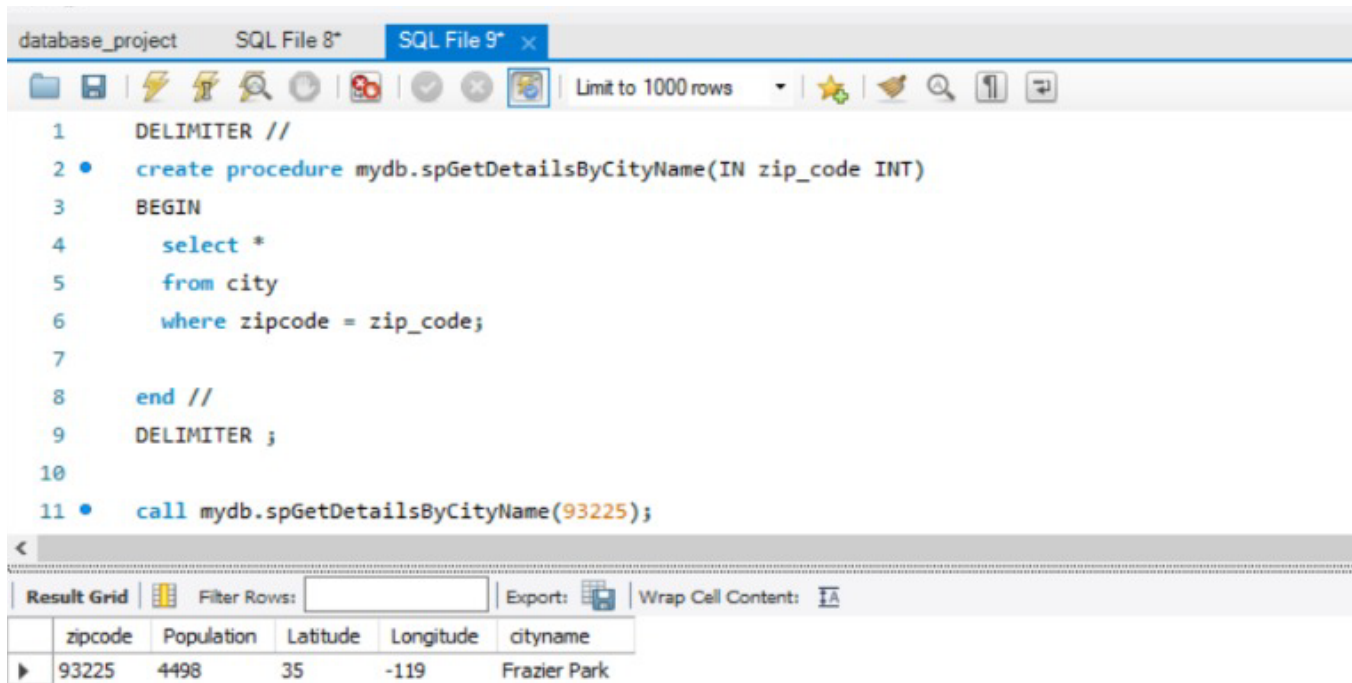
(4)If you did not use MySql Workbench on Windows, describe the exact software/hardware platform you used

We used MySql Workbench on Windows.

(5)Provide a brief users guide describing in sufficient detail what each View/Stored procedure does in English.
View : In SQL, a view is a virtual table based on the result-set of an SQL statement.A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.Unlike stored procedure , does not accept parameters.

Stored procedure : A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

(6)Include the outputs of your favorite/interesting/challenging stored procedures(with IN, OUT and INOUT)/views
IN Stored Procedure :



```
1 DELIMITER //
2 create procedure mydb.spGetDetailsByCityName(IN zip_code INT)
3 BEGIN
4     select *
5     from city
6     where zipcode = zip_code;
7
8 end //
9 DELIMITER ;
10
11 call mydb.spGetDetailsByCityName(93225);
```

zipcode	Population	Latitude	Longitude	cityname
93225	4498	35	-119	Frazier Park

OUT Stored Procedure:

```
database_project  SQL File 8*  SQL File 9* x
Limit to 1000 rows
12
13 DELIMITER //
14 • create procedure mydb.spGetAvgGbDownload(OUT avgGb INT)
15 BEGIN
16     select avg(value_integer)
17     INTO avgGb
18     from customer_has_property_integer
19     where property_integer_idproperty_integer = (select idproperty_integer from property_integer where property_integercol_name="Avg Monthly GB Download");
20 end //
21 DELIMITER ;
22
23 • call mydb.spGetAvgGbDownload(@avgGb);
24 • select @avgGb;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	@avgGb
▶	26

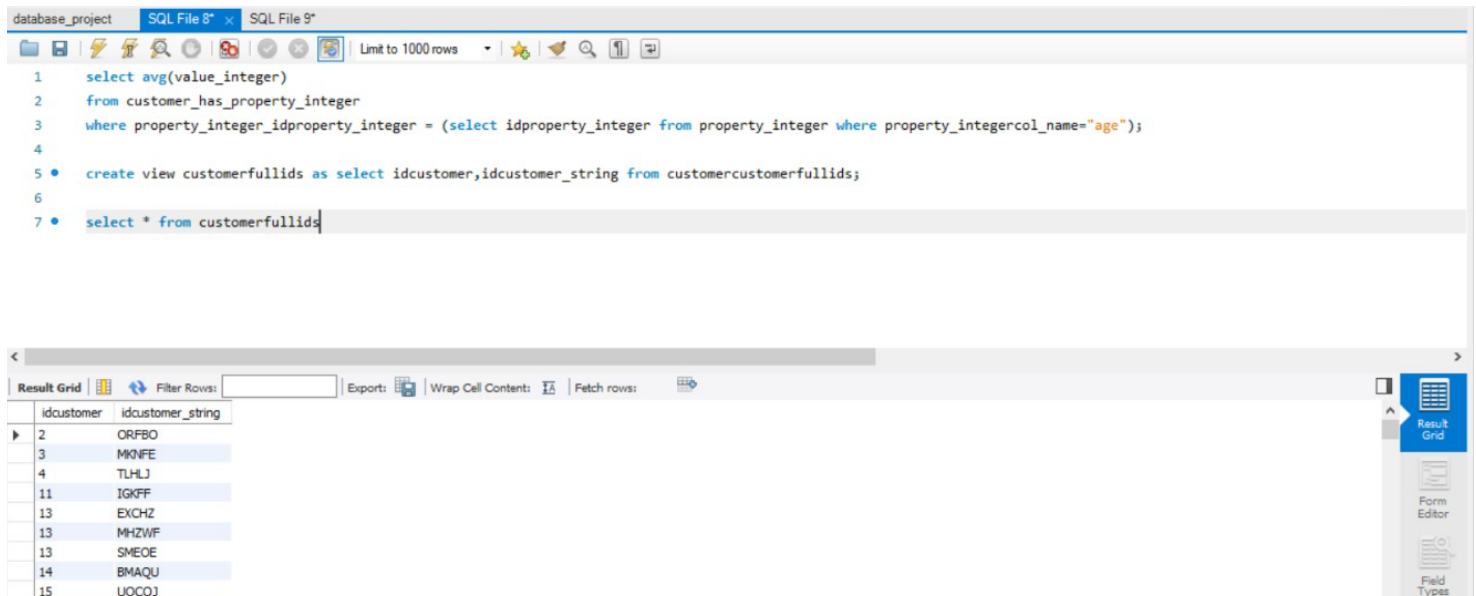
INOUT Stored Procedure:

```
database_project  SQL File 8*  SQL File 9* x
Limit to 1000 rows
25
26
27 DELIMITER //
28 • create procedure mydb.spNewTotalCharges(INOUT total_charge float, IN kdv float)
29 BEGIN
30     set total_charge = total_charge + kdv;
31 end //
32 DELIMITER ;
33
34 • set @total_charge = 15.87;
35
36 • call mydb.spNewTotalCharges(@total_charge,12);
37 • select @total_charge;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	@total_charge
▶	27.869998931884766

View:



(7) Give an analysis of your system's limitations and list suggested possibilities for improvement

Columns in our system are almost all customer properties, if we could separate these columns independently from the customer, our sql diagram would be 3nf instead of rdf.

(8) Submit a full relational table specification of your database in the SQL Database Definition Language (DDL) This specification should include both the data type of each attribute the not null constraint when appropriate and sample data values for each attribute represented as comments You should also specify the primary keys (e g primary key (ssn)) and referential constraints (e g foreign key (mgrssn) references employee(ssn)) Many groups included this specification in their project proposal.

```

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- Table `city`
--
DROP TABLE IF EXISTS `city` ;

CREATE TABLE IF NOT EXISTS `city` (
  `zipcode` INT NOT NULL,
  `Population` INT NOT NULL,
  `Latitude` DECIMAL NOT NULL,
  `Longitude` DECIMAL NOT NULL,
  `cityname` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`zipcode`))
ENGINE = InnoDB;

-- Table `customer`

```

```
-- -----  
DROP TABLE IF EXISTS `customer` ;
```

```
CREATE TABLE IF NOT EXISTS `customer` (  
  `idcustomer` INT NOT NULL,  
  `idcustomer_string` VARCHAR(45) NOT NULL,  
  `city_zipcode` INT NOT NULL,  
  PRIMARY KEY (`idcustomer`, `idcustomer_string`))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `columns`  
-- -----
```

```
DROP TABLE IF EXISTS `columns` ;
```

```
CREATE TABLE IF NOT EXISTS `columns` (  
  `idcolumns` INT NOT NULL,  
  `columns_name` VARCHAR(45) NOT NULL,  
  `columns_type` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idcolumns`))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `column_integers`  
-- -----
```

```
DROP TABLE IF EXISTS `column_integers` ;
```

```
CREATE TABLE IF NOT EXISTS `column_integers` (  
  `columns_idcolumns` INT NOT NULL,  
  `customer_idcustomer` INT NOT NULL,  
  `customer_idcustomer_string` VARCHAR(45) NOT NULL,  
  `value` INT NULL,  
  PRIMARY KEY (`columns_idcolumns`, `customer_idcustomer`, `customer_idcustomer_string`))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `column_varchar`  
-- -----
```

```
DROP TABLE IF EXISTS `column_varchar` ;
```

```
CREATE TABLE IF NOT EXISTS `column_varchar` (  
  `columns_idcolumns` INT NOT NULL,  
  `customer_idcustomer` INT NOT NULL,  
  `customer_idcustomer_string` VARCHAR(45) NOT NULL,  
  `value` VARCHAR(45) NULL,  
  PRIMARY KEY (`columns_idcolumns`, `customer_idcustomer`, `customer_idcustomer_string`))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `column_float`  
-- -----
```

```
DROP TABLE IF EXISTS `column_float` ;
```

```
CREATE TABLE IF NOT EXISTS `column_float` (  
  `columns_idcolumns` INT NOT NULL,  
  `customer_idcustomer` INT NOT NULL,  
  `customer_idcustomer_string` VARCHAR(45) NOT NULL,  
  `value` FLOAT NULL,  
  PRIMARY KEY (`columns_idcolumns`, `customer_idcustomer`, `customer_idcustomer_string`))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `column_tinyint`  
-- -----
```

```
DROP TABLE IF EXISTS `column_tinyint` ;
```

```
CREATE TABLE IF NOT EXISTS `column_tinyint` (  
  `columns_idcolumns` INT NOT NULL,  
  `customer_idcustomer` INT NOT NULL,  
  `customer_idcustomer_string` VARCHAR(45) NOT NULL,  
  `value` TINYINT(1) NULL,  
  PRIMARY KEY (`columns_idcolumns`, `customer_idcustomer`, `customer_idcustomer_string`))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `property_integer`  
-- -----
```

```
DROP TABLE IF EXISTS `property_integer` ;
```

```
CREATE TABLE IF NOT EXISTS `property_integer` (  
  `idproperty_integer` INT NOT NULL,  
  `property_integercol_name` VARCHAR(45) NULL,  
  PRIMARY KEY (`idproperty_integer`))  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `property_varchar`  
-- -----
```

```
DROP TABLE IF EXISTS `property_varchar` ;
```

```
CREATE TABLE IF NOT EXISTS `property_varchar` (  
  `idproperty_varchar` INT NOT NULL,  
  `property_vacharcol_name` VARCHAR(45) NULL,  
  PRIMARY KEY (`idproperty_varchar`))  
ENGINE = InnoDB;
```

```

-- -----
-- Table `property_float`
-- -----
DROP TABLE IF EXISTS `property_float` ;

CREATE TABLE IF NOT EXISTS `property_float` (
  `idproperty_float` INT NOT NULL,
  `property_floatcol_name` VARCHAR(45) NULL,
  PRIMARY KEY (`idproperty_float`))
ENGINE = InnoDB;

-- -----
-- Table `property_tinyint`
-- -----
DROP TABLE IF EXISTS `property_tinyint` ;

CREATE TABLE IF NOT EXISTS `property_tinyint` (
  `idproperty_tinyint` INT NOT NULL,
  `property_tinyintcol_name` VARCHAR(45) NULL,
  PRIMARY KEY (`idproperty_tinyint`))
ENGINE = InnoDB;

-- -----
-- Table `customer_has_integer_values`
-- -----
DROP TABLE IF EXISTS `customer_has_integer_values` ;

CREATE TABLE IF NOT EXISTS `customer_has_integer_values` (
  `idcustomer_has_integer_values` INT NOT NULL)
ENGINE = InnoDB;

-- -----
-- Table `customer_has_property_integer`
-- -----
DROP TABLE IF EXISTS `customer_has_property_integer` ;

CREATE TABLE IF NOT EXISTS `customer_has_property_integer` (
  `customer_idcustomer` INT NOT NULL,
  `customer_idcustomer_string` VARCHAR(45) NOT NULL,
  `property_integer_idproperty_integer` INT NOT NULL,
  `value_integer` INT NULL,
  PRIMARY KEY (`customer_idcustomer`, `customer_idcustomer_string`,
`property_integer_idproperty_integer`))
ENGINE = InnoDB;

-- -----
-- Table `customer_has_property_varchar`

```

```

-- -----
DROP TABLE IF EXISTS `customer_has_property_varchar` ;

CREATE TABLE IF NOT EXISTS `customer_has_property_varchar` (
  `customer_id` INT NOT NULL,
  `customer_id_string` VARCHAR(45) NOT NULL,
  `varchar_id` INT NOT NULL,
  `value` VARCHAR(45) NULL,
  PRIMARY KEY (`customer_id`, `customer_id_string`, `varchar_id`))
ENGINE = InnoDB;

-- -----
-- Table `customer_has_property_float`
-- -----
DROP TABLE IF EXISTS `customer_has_property_float` ;

CREATE TABLE IF NOT EXISTS `customer_has_property_float` (
  `customer_idcustomer` INT NOT NULL,
  `customer_idcustomer_string` VARCHAR(45) NOT NULL,
  `property_float_idproperty_float` INT NOT NULL,
  `value_float` FLOAT NULL,
  PRIMARY KEY (`customer_idcustomer`, `customer_idcustomer_string`,
`property_float_idproperty_float`))
ENGINE = InnoDB;

-- -----
-- Table `customer_has_property_tinyint`
-- -----
DROP TABLE IF EXISTS `customer_has_property_tinyint` ;

CREATE TABLE IF NOT EXISTS `customer_has_property_tinyint` (
  `customer_idcustomer` INT NOT NULL,
  `customer_idcustomer_string` VARCHAR(45) NOT NULL,
  `property_tinyint_idproperty_tinyint` INT NOT NULL,
  `values_tinyint` TINYINT NULL,
  PRIMARY KEY (`customer_idcustomer`, `customer_idcustomer_string`,
`property_tinyint_idproperty_tinyint`))
ENGINE = InnoDB;

```

(9) Include all your SQL code used in your system as well as any additional Php/Perl/Python/Java/SQL-Loader programs you used for data acquisition and input to your presentation. If you have long sequences of input statements in excess of 2 pages provide a 2-3 page representative sample.

We uploaded all SQL codes and python codes to the Bitbucket.

