

|  |
| --- |
| 4005-cem Resit report |
|  |
| July 4  Student Id: 11782139  Authored by: talha asher |



### Index

## [A-](#_A=) [An answer to the following question, together with a justification for the decision: In its current form, it is a traditional database. Should you keep it that way?](#_A=)

## [B- The detailed process of implementing the database using SQL, including the normalization of the table (should that be required), the identification of the attributes, the Entity-Relationship Diagram, and the use of SQL commands to create and populate the tables with data.](#_B=)

## [C-1. Find the supervisors of the branches which rented out SUVs – output the supervisor’s ID and name.](#_C.1=)

## [C-2. Find the supervisors of the branches which had bills higher than 500 – output the supervisor’s ID and name.](#_C.2=)

## [C-3. Find the supervisors of the branches which rented out cars in May 2021 – output the supervisor’s ID and name.](#_C-3._Find_the)

## [C-4. Find the supervisors of the branches which handed out penalties – output the supervisor’s ID and name.](#_C.4=)

## [C-5. Find the type of car that was rented the highest number of times – output the car type and the number of times.](#_C.5=)

## [D-A choice of 4 types of graphs suitable for analysing the information in the database, their Implementation in Python (both the code and the output graph itself should be included in the report) and an explanation of what the graphs illustrate in your own words](#_D=)

## References

## [A](#_A-_An_answer)=

Normalization is a technique for organizing data in a database. It is important that a database is normalized as it minimizes redundancy and to ensure only related data is stored in each table. Also, it helps as Updates run quickly due to no data being duplicated in multiple locations, inserting data run quickly since there is only a single insertion point for a piece of data and no duplication is required, Tables are typically smaller than the tables found in non-normalized databases. This usually allows the tables to fit into the buffer, thus offering faster performance. As compared to unnormalized database there is more redundant data therefore updates are slower when inserting

data u does not need to worry about duplicate data and worse performance overall.

## [B=](#_B-_The_detailed)

Table

Description automatically generated

1nf:

Graphical user interface, application, table, Excel

Description automatically generated

Bill nr --PK and FK bill nr – PK

2nf: Graphical user interface, application

Description automatically generated

Bill nr --PK and FK branch code --PK bill nr-- PK

Branch code –FK

3nf:

Bill nr -- PK and FK bill nr--fk, branch code--fk and car late nr-- fk

Bill nr—pK car\_plate\_nr --pk

Table

Description automatically generated Table

Description automatically generated Table

Description automatically generated

Bill detail linker table car detail

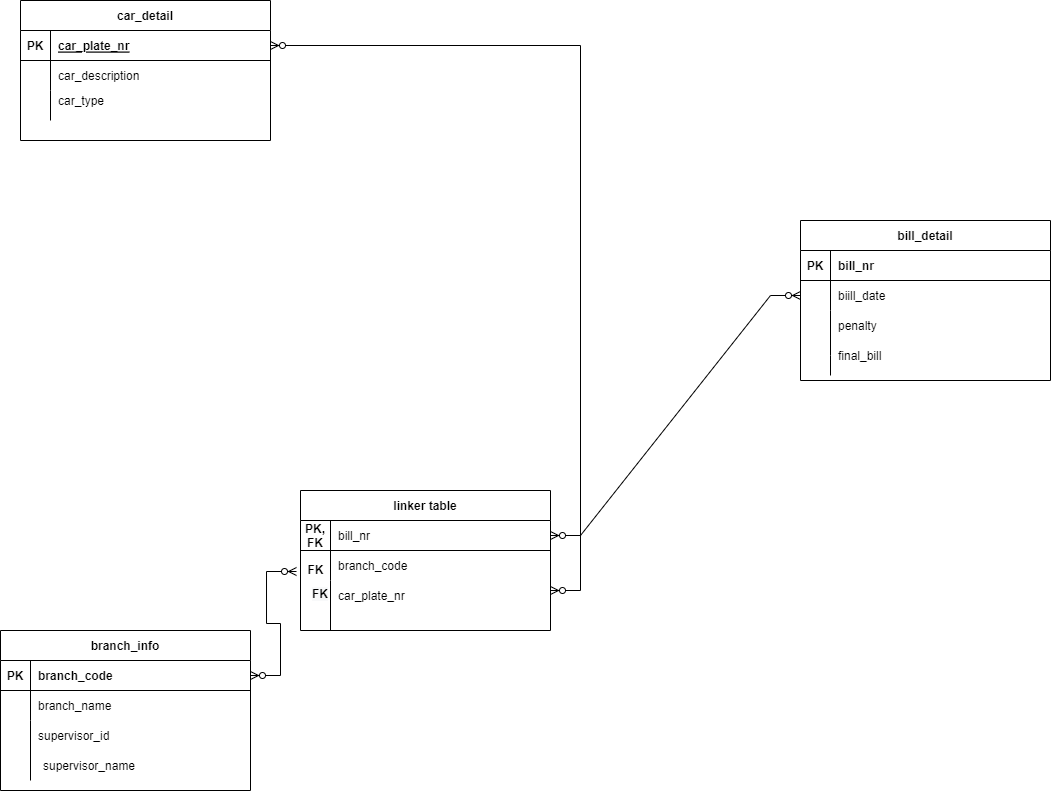
Table

Description automatically generated

Branch info

Branch code -- PK

Fist of all I split table in two with bill nr as primary key then I took branch name ,banch id ,supervsior id ,superviosr name in seprate table with branch code as priamry key , then linked bill nr and branch code and put them in one table and lastly I seprtaed car plate nr(PK) ,car description and car stype in seprate tabel and put bill nr(PK) brancah code and car plate nr togather in table so everthing links up with each other.



I created the entity relation diagram with many to many relations. Car\_detail , bill\_detail, branch \_info has many to many relations to linker\_table.

Then I created a database through SQLite studio 3 and used SQL command to create table and insert data into table.

CREATE TABLE IF NOT EXISTS branch\_info ("branch\_code" INTEGER,"branch\_name" CHAR,"supervisor\_id"INTEGER,"supervisor\_name" CHAR,PRIMARY KEY("branch\_code"));

INSERT INTO branch\_info ( branch\_code, branch\_name, supervisor\_id, supervisor\_name )

                    VALUES

                    (100023,'Coventry',871,'Anna Smith'),

                    (456109,'Leamington Spa',149,'John Cruise'),

                    (555901,'Wolverhampton',111,'Catherine Johnson'),

                    (876734,'Walsall',102,'David Brown'),

                    (981256,'Warwick',823,'James Doherty');

Firstly, I used I created table using CARETE table SQL command and named table branch info and then made column such as branch code (integer) which is PRNAMRY KEY ,branch name (char) ,supervisor id (integer), supervisor name (char). After that I inserted data in table using INSERT command .

CREATE TABLE IF NOT EXISTS car\_detail (car\_plate\_nr VARCHAR, car\_description VARCHAR, car\_type CHAR, PRIMARY KEY (car\_plate\_nr));

INSERT INTO car\_detail ( car\_plate\_nr, car\_description, car\_type)

                VALUES

                ('DS4049','BA1234','SUV'),

                ('DL3434','BA6753','Sports\_Car'),

                ('OP9817','BA1561','SUV'),

                ('SJ7182','BA9878','Hatchback'),

                ('BN9745','BA9123','SUV'),

                ('LA5142','BA8177','Sedan'),

                ('CB0098','BA4545','Sports\_Car'),

                ('ZX7222','BA1000','Coupe'),

                ('QW0128','BA8882','Sedan'),

                ('PO8123','BA5656','SUV'),

                ('IU7878','BA0012','Hatchback'),

                ('GF5612','BA3421','Sedan'),

                ('NM8787','BA4545','Sports\_Car'),

                ('VC1111','BA8177','Sedan'),

                ('FG7100','BA9123','Hatchback'),

                ('RE6000','BA9878','Sedan'),

                ('TR6199','BA1561','SUV'),

                ('DR1166','BA6753','Sports\_Car'),

                ('BP9111','BA1234','Coupe');

Similarly, to before I used to create table SQL command to create command with column car plate nr (varchar ) which Is PK , car description (varchar) and car type (char) and inserted data using INSERT command.

CREATE TABLE IF NOT EXISTS bill\_detail (bill\_nr INTEGER, bill\_date DATE, penalty INTEGER, final\_bill INTEGER, PRIMARY KEY (bill\_nr));

INSERT INTO bill\_detail ( bill\_nr, bill\_date,penalty, final\_bill)

                VALUES

                (166651, '18/01/2021',50,1050),

                (123111, '19/02/2021',0,500),

                (561909, '06/03/2021',0,480),

                (565690, '29/01/2021',0,680),

                (128976, '10/10/2021',0,710),

                (511899, '25/11/2021',20,1500),

                (141421, '03/12/2021',0,850),

                (514879, '29/10/2021',0,1250),

                (771100, '16/11/2021',20,300),

                (675912, '06/01/2022',50,350),

                (991762, '08/02/2022',0,950),

                (110054, '19/07/2021',100,1400),

                (378123, '12/08/2021',20,450),

                (808051, '18/09/2021',0,670),

                (100023, '21/07/2021',0,1030),

                (611554, '27/08/2021',50,520),

                (888712, '10/04/2021',0,490),

                (343412, '28/05/2021',20,1230),

                (222678, '04/06/2021',0,1680);

Similarly, to before I used to create table SQL command to create command with column bill nr(integer) which Is PK , car description (varchar) and bill date (DATE), penalty (integer) , final bill (integer ) and inserted data using INSERT command .

CREATE TABLE IF NOT EXISTS linker\_table (bill\_nr INTEGER PRIMARY KEY REFERENCES bill\_detail (bill\_nr) MATCH "FULL", branch\_code INTEGER REFERENCES branch\_info (branch\_code) MATCH SIMPLE, car\_plate\_nr VARCHAR REFERENCES car\_detail (car\_plate\_nr) MATCH "FULL");

INSERT INTO linker\_table  ( bill\_nr, branch\_code, car\_plate\_nr)

                VALUES

                (166651,876734, 'DS4049'),

                (123111,876734, 'DL3434'),

                (561909,876734, 'OP9817'),

                (565690,876734, 'SJ7182'),

                (128976,100023, 'BN9745'),

                (511899,100023, 'LA5142'),

                (141421,100023, 'CB0098'),

                (514879,100023, 'ZX7222'),

                (771100,100023, 'QW0128'),

                (675912,456109, 'PO8123'),

                (991762,456109, 'IU7878'),

                (110054,981256, 'GF5612'),

                (378123,981256, 'NM8787'),

                (808051,981256, 'VC1111'),

                (100023,981256, 'FG7100'),

                (611554,981256, 'RE6000'),

                (888712,555901, 'TR6199'),

                (343412,555901, 'DR1166'),

                (222678,555901, 'BP9111');

Lastly, like before I used to create table SQL command to create command with column bill nr(integer) which Is PK and FK, branch code (integer) Fk and car plate nr (varchar) and inserted data using INSERT command , with REFERENCES command to link with primary key of other table.

## [C.1=](#_C-1._Find_the)

SELECT branch\_info.supervisor\_id, branch\_info.supervisor\_name from car\_detail

join linker\_table on car\_detail.car\_plate\_nr = linker\_table.car\_plate\_nr

join branch\_info on linker\_table.branch\_code = branch\_info.branch\_code

where car\_detail.car\_type = "SUV":

For this query to select supervisor id and name from branch info and used join statement to link car\_plate\_nr from car\_detail to car\_plate\_nr in linker table and branch\_code from linker\_table to branch code in branch info table and finally used where statement to search for SUV in car type.

This gives me supervisor id and supervisor name of benches that rented out SUV.

## [C.2=](#_C-2._Find_the)

SELECT branch\_info.supervisor\_id, branch\_info.supervisor\_name

from bill\_detail

join linker\_table on bill\_detail.bill\_nr = linker\_table.bill\_nr

join branch\_info on linker\_table.branch\_code = branch\_info.branch\_code

where bill\_detail.final\_bill >500;

For this query to select supervisor id and name from branch info and used join statement to link bill\_nr from bill\_detail to bill\_nr in linker table and branch\_code from linker\_table to branch code in branch info table and finally used where statement to search for final bill over 500.

This gives me supervisor id and supervisor name of branches that had final bill over 500.

## [C.3=](#_C-3._Find_the)

SELECT branch\_info.supervisor\_id, branch\_info.supervisor\_name

from bill\_detail

join linker\_table on bill\_detail.bill\_nr = linker\_table.bill\_nr

join branch\_info on linker\_table.branch\_code = branch\_info.branch\_code

where bill\_detail.bill\_date>='01/05/2021' AND bill\_detail.bill\_date <= '31/05/2021';

For this query to select supervisor id and name from branch info and used join statement to link bill\_nr from bill\_detail to bill\_nr in linker table and branch\_code from linker\_table to branch code in branch info table and finally used where statement to search for branches who rented out car in month of May.

This gives me supervisor id and supervisor name of branches that rented out car in month of May .

## [C.4=](#_C-4._Find_the)

SELECT branch\_info.supervisor\_id, branch\_info.supervisor\_name

from bill\_detail

join linker\_table on bill\_detail.bill\_nr = linker\_table.bill\_nr

join branch\_info on linker\_table.branch\_code = branch\_info.branch\_code

where bill\_detail.penalty>0;

For this query to select supervisor id and name from branch info and used join statement to link bill\_nr from bill\_detail to bill\_nr in linker table and branch\_code from linker\_table to branch code in branch info table and finally used where statement to search for branches that gave penalty.

This gives me supervisor id and supervisor name of branches that gave penalty .

## [C.5=](#_C-5._Find_the)

SELECT car\_detail.car\_type, count(car\_detail.car\_type) as number\_of\_times

FROM car\_detail, linker\_table

where car\_detail.car\_plate\_nr = linker\_table.car\_plate\_nr

GROUP BY car\_type

ORDER BY number\_of\_times desc

LIMIT 2;

For this query I used select statement to all car\_type and used count statement to count how many times each car type is repeated , then I linked car\_palte\_nr in car detail to car plate nr to linker table.

I used group statement to group car type together and then used order statement to order result in descending order and used limit statement 2 as two car type were used equal amount of times.

This gives me car type most rented with how many times it was rented.

## [D=](#_D-A_choice_of) import pandas as pd

import matplotlib

data=pd.read\_csv(r"C:/Users/Talha/Desktop/R/4005/work..csv")

%matplotlib inline

data["Car\_Type"].value\_counts().plot(kind="bar")

Chart, bar chart

Description automatically generated

I would recommend using bar chart to figure what is most rented out vehicle as then u can invest in those vehicles for more profit as more the car that people want to rent will be available.

import pandas as pd

import matplotlib

data=pd.read\_csv(r"C:/Users/Talha/Desktop/R/4005/work..csv")

%matplotlib inline

data["Penalty"].value\_counts().plot(kind="pie",autopct="%1.1f%%")

Chart, pie chart

Description automatically generated

I would recommend using pie chart to figure how much penalty people are giving so we can investigate about it and figure out if they have been penalty for no reason.

Pie chart to figure out how much penalty are they paying.

## References:

<https://www.w3schools.com/python/python_mysql_getstarted.asp>

<https://realpython.com/python-sql-libraries/>

<https://www.geeksforgeeks.org/sql-using-python/>

<https://builtin.com/data-science/how-to-use-sql-python>

<https://swcarpentry.github.io/sql-novice-survey/10-prog/index.html>

<https://www.analyticsvidhya.com/blog/2021/06/plotting-visualizations-out-of-pandas-dataframes/>