```
/*
This is how you write the comments. Comments are
the text that won't get executed
*/

/*Tasks that we'll perform right now.
1. Create the database - loan_db
2. Create the tables in the database.
- Create columns in each table
- Assign data types (Type of data a column can store)
(4 imp data types:
: Number / Integer
: Character / Text
: Date / Time
: Binary / Boolean (yes / no)
- Create constraints (like PK, FK, Unique, etc.)
- Create relationship constraints between tables

3. How to see the relationship diagram (Entity Relationship Diagram)
a. Go to Database
b. Select Reverse Engineer (Shortcut: Ctl + R)
c. Then follow the on screen steps
*/

/*1. Create loan_db
All SQL commands should end with a semi colon
SQL is not case sensitive, except for some functions.*/

drop database loan_db_new;

CREATE DATABASE loan_db_new;

/*2. Create the tables in the database.
- Create columns in each table
- Assign data types (Type of data a column can store)
(4 imp data types:
: Number / Integer
: Character / Text (alphanumeric, including special characters)
: Date / Time / DateTime
: Binary / Boolean (yes / no; 0/1)
- Create constraints (like PK, FK, Unique, etc.)
- Create relationship constraints between tables
*/

use loan_db_new; /*selects the loan_db_new database (makes it default)*/

CREATE TABLE Customers
(CustID INT Primary Key,
CustName Char(100), /*Static memory allocation*/
Email VarChar(50) unique, /*Dynamic memory allocation*/
DOB Date NOT NULL
```

```
);

describe customers;

alter table customers
drop primary key;

alter table customers
add constraint primary Key(custID);

create table employee
(empId int primary key,
FirstName varchar(100),
lastName varchar(100),
email varchar(100) Not null,
Salary int default 0
);

CREATE TABLE Loans
(LoanID INT,
LoanStatus binary, /*True or False*/
LoanType varchar(100),
EmpID INT,
constraint L_pk Primary key(loanID),
constraint L_fk1 foreign key(empID)
   references employee(empID)
);

create table cust_loans(
custId INT,
loanID INT,
constraint cl_pk Primary key(custId, loanID),
constraint cl_fk1 foreign key(custId)
   references customers(custID),
constraint cl_fk2 foreign key(loanID)
   references loans(loanID)
);
Alter table loanstatus
drop foreign key l_fk2;


alter table loanstatus
drop foreign key l_fk1;

alter table loanstatus
add constraint l_fk1 FOREIGN KEY(custID)
REFERENCES customers(CustID);

ALTER TABLE Orders
ADD Constraint o_pk Primary KEY(OID);
```

```
/*Copying the customers table into a new customers2 table
without the data*/
create table customers2
select *
from customers;

select * from customers2;

alter table customers2
add column temp int;

alter table customers2 rename column temp to CreditScore;

select * from customers2;

/*How do we add data to a new column?
Let's CreditScore =
if(gender = "Female",(ApplicantIncome*.1)*(Dependents/2),
(ApplicantIncome*.08)*(dependents/2))
*/
update customers2
SET creditScore = if(gender =
"female",(ApplicantIncome*.1)*(Dependents/2),(ApplicantIncome*.08)*(dependents/2));

select custID, gender, creditscore, applicantIncome
from customers2;

/*different values in the new column*/
update customers
SET creditScore = CASE
                WHEN (gender = "female" AND dependents > 0) THEN 999
                WHEN (gender = "female" and dependents > 2) THEN 1200
                When (gender = "male" and dependents > 0) THEN 899
                WHEN (gender = "male" and dependents > 2) THEN 1099
                ELSE 0
                End;

/*Create a new column customers table -> RiskProfile with
data type text. The value of the RiskProfile will be as per this logic:
- Risky when a customer is not self employed
   and applicantincome < 2000
- Moderate Risk when a customer is not self employed
   and applicantincome > 2000
- Not Risky when a customer is self employed and
      applicantIncome > 4000
*/
alter table customers2 add column RiskProfile char(100);

/*---------------------------------------
How to migrate the table's data from one database to another?
We'll do the following now:
```

```
1. Copy the design of the customers table from loans_db_new's customers table and
create a new table in a new database (let's say loan_temp database).
2. Copy the data from customers table from the old db loans_db_new
to the new db's customers table.
*/
create table loan_new_oct.customers3
select *
from loans_db_new.customers; /*since we've given an impossible condition, this query will
just
        return the columns without any rows from the customers table.*/

/*We'll now use create statement along with select statement which only returns
columns to copy the columns to a new table in another database*/
create table loans_temp.customers
select *                /*this select statement will only return the columns*/
from loans_new_db.customers
where 1>2;

select *
from loans_temp.customers;

/*How to copy the data from one table of a database to a table of another database?
Extract the data from the old loan_db_new db's customers table and
insert them into loan_temp's customers table*/
INSERT INTO loan_temp.Customers
SELECT *
FROM loan_db_new.customers;


/*-----------------------------------------------
 NEW DATABASE SCRIPTS (OPTIONAL to run these)
  -----------------------------------------------*/
/*The following scripts are to create a new database called officestationary.
Feel free to explore, but not mandatory to run these scripts.*/

create database officestationary;

use officestationary;

CREATE TABLE Customers
(CustID INT,
CustName Char(100),
CustCity VarChar(100),
DOB Date,
CustNetWorth INT,
Constraint cust_pk PRIMARY KEY(CustID)
);

CREATE TABLE Orders
(OID INT,
Order_Date date,
```

```
Order_ShipDate date,
CID INT,
Constraint o_fk FOREIGN KEY(CID) REFERENCES customers(CustID)
);

ALTER TABLE Orders
ADD Constraint o_pk Primary KEY(OID);

CREATE TABLE Products
(PID INT,
PName VarChar(100) NOt NULL,
Price INT default 0,
Constraint p_pk PRIMARY KEY(PID),
Constraint p_unq UNIQUE(PName),
Constraint p_chk Check(Price > 0)
);

/*Junction table - OrderProduct*/
CREATE TABLE Order_Product
(OID INT,
PID INT,
Constraint op_pk PRIMARY KEY(oid, pid), /*Composite PK*/
Constraint op_fk1 Foreign Key(OID) REFERENCES ORders(OID), /*FK*/
Constraint op_fk2 Foreign Key(PID) REFERENCES Products(PID) /*FK*/
);

/*
alter table <nameOfTheTable>
<ADD / DROP / Modify / Rename> <Column / Constraint / Primary key /
Foreign Key referenceTag / check referencetag>; */

ALTER TABLE Orders
DROP Foreign Key o_fk;

ALTER TABLE Orders
ADD constraint o_fk Foreign KEy(CID)
REFERENCES customers(CustID);

ALTER TABLE Orders
DROP Foreign Key o_fk;

/*Drop the primary key constraint. Don't drop the column. */
ALTER TABLE customers
DROP Primary Key;

/*Drop the column custID*/
ALTER TABLE customers
DROP Column custID;

/*simple example -
its like break-up .....but the person is still there*/
```

```
ALTER TABLE Customers
ADD Constraint cust_pk PRIMARY KEY(CustID);

ALTER TABLE Products
Drop index p_unq;

/*Make custName column as non-unique (can have repetition)*/
ALTER TABLE Customers
DROP index cust_unq;

/*Add the unique constraint again to the customer table*/
ALTER TABLE Customers
ADD Constraint cust_unq Unique(CustName);

/*In Order Product table, drop the fK for Product table*/
ALTER TABLE Order_PRoduct
DROP foreign key op_fk3;

ALTER TABLE order_product
ADD constraint op_fk2 Foreign Key(PID) REFERENCES Products(PID);

/*How to drop a composite primary key*/
ALTER TABLE order_product
DROP primary key;

/*Drop the city column from the customer table*/
ALTER TABLE Customer
DROP Column City;

/*Add the column city in the customre table*/
ALTER TABLE Customer
ADD Column city VarChar(100);

/*Rename city column to Cust_City*/
ALTER TABLE customer
Rename column City to Cust_City;

/*Rename the customer table to customers*/
Alter Table Customer Rename Customers;

ALTER TABLE customers
Modify Column cust_city Char(250);

ALTER TABLE customers
Modify Column cust_city VarChar(100);

Alter TABLE Customers
Add column networth INT;

ALTER TABLE Customers
```

```
Add column country enum('India', 'USA', 'Germany');

DROP Table Order_Product;

ALTER TABLE Customers ADD Column DOB Date;

use officestationary_db2;

/*Insert rows into customers table
Default date format supported by MySQL yyyy-mm-dd */
INSERT INTO Customers(CustID, CustName, CustCity, DOB)
VALUES(101, "Puja", "Mumbai", "1990-12-05");

SELECT * FROM Customers; /*Helps us query / extract the data from the table*/

INSERT INTO customers(CustID, CustName, CustCity, DOB)
Values
(401, "Reliance", "Mumbai", '1989-11-23'),
(201, "Tata Sons", "Mumbai", '1978-09-30'),
(301, "Bharti Airtel", "Delhi",'1999-01-22');

SELECT *
FROM Customers;

/*Deletes all the rows of a table. The table will remain, only the rows will get deleted.*/
truncate customers;

Delete
from customers
where custcity = "Mumbai";

create database ofs_sept;

/*
1. Copy the design of the customers table from ofs_db2's customers table
2. Copy the data from customers table from ofs_db2 to the new db's customers table.
*/

select *
from officestationary_db2.customers
where 1>2;

select *
from officestationary_db2.customers
where 1>2;


create table ofs_sept.customers
select *
from officestationary_db2.customers
where 1>2;
```

```
select *
from ofs_sept.customers;

drop table ofs_sept.customers;

/*Extract the data from office_stationary db's customers table
Insert them into office_stationary_db2's customers table*/
INSERT INTO Customers(CustID, CustName, CustCity)
SELECT custID, CustName, CustCity
FROM office_stationary.customers;

SELECT * FROM Customers;

/*Valid date format for MySQL is yyyy-mm-dd. Eg: 2020-04-12*/
INSERT INTO Orders(OID,OrderDate,OrderShipDate,CID)
Values(1,'2020-04-08','2020-04-15', 201);

SELECT * FROM office_Stationary.Customers;

/*How do we modify the existing data?*/
Update customers
Set Custcity = "Delhi"
WHERE custCity = "Indraprastha";

Update customers
SET CustName = "Idea", CustCity = "Kolkata"
WHERE CustID = 101;

SELECT * FROM Customers;

SELECT *
FROM orders;

DELETE
FROM Customers;

TRUNCATE Customers;
/*By default, you're not allowed to run delete or update statemetns
without a where clause (this is called as safe updates / deletes)*/

SET SQL_SAFE_UPDATES = 0;
SET SQL_SAFE_UPDATES = 1;

SELECT * FROM Customers;

/*Delete all the rows from the orders table.
Similar to delete without any where condition*/
Truncate Orders;

Create TABLE Employees
```

```
(EmpID INT auto_increment,
EmpName VarChar(100),
Constraint emp_pk PRIMARY KEY(EmpID)
);

INSERT INTo Employees(EmpName)
VALUES("Divya");

Select * FROM Employees;
```