```
drop schema if exists lab6a;

# 1
/*
Skapa databas
Utifrån givet ERD (nedan) skapa en databas med tabeller med datatyper och
constraints som passar.
 Eventuellt lä'g till id som primary key där det passar och/eller behövs.
 Redovisa alla queries som behövs för att skapa databasen.
 Du får byta namn på och lägga till attribut om du vill (mot vad som finns
i ERD).
 Funktionalitet ska dock vara enligt specifikationerna (men namn på
parametrar och
 procedures med mera kan du döpa så som du tycker är bra).
 */
-- Baserad på ERD-Modell verktyget i MySQL Workbench

-- MySQL Workbench Forward Engineering

-- -----------------------------------------------------
-- Schema lab6a
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `lab6a` DEFAULT CHARACTER SET utf8mb4;
USE `lab6a`;

-- -----------------------------------------------------
-- Table `lab6a`.`Authors`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `lab6a`.`Authors`
(
    `idAuthors` INT         NOT NULL AUTO_INCREMENT,
    `name`      VARCHAR(32) NULL,
    `username`  VARCHAR(16) NULL,
    UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE,
    PRIMARY KEY (`idAuthors`)
)
    ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `lab6a`.`Posts`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `lab6a`.`Posts`
(
    `idPosts`           INT          NOT NULL AUTO_INCREMENT,
    `content`           VARCHAR(255) NULL,
    `subject`           VARCHAR(32)  NULL,
    `posted`            DATETIME     NULL,
    `last_edit`         DATETIME     NULL,
    `Authors_idAuthors` INT          NOT NULL,
    PRIMARY KEY (`idPosts`),
    INDEX `fk_Posts_Authors_idx` (`Authors_idAuthors` ASC) VISIBLE,
    CONSTRAINT `fk_Posts_Authors`
        FOREIGN KEY (`Authors_idAuthors`)
```

```
            REFERENCES `lab6a`.`Authors` (`idAuthors`)
            ON DELETE CASCADE
            ON UPDATE CASCADE
)
    ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `lab6a`.`Categories`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `lab6a`.`Categories`
(
    `idCategories` INT         NOT NULL AUTO_INCREMENT,
    `name`         VARCHAR(16) NULL,
    PRIMARY KEY (`idCategories`)
)
    ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `lab6a`.`Comments`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `lab6a`.`Comments`
(
    `idComments`    INT          NOT NULL AUTO_INCREMENT,
    `name`          VARCHAR(16)  NULL,
    `subject`       VARCHAR(32)  NULL,
    `content`       VARCHAR(255) NULL,
    `posted`        DATETIME     NULL,
    `Posts_idPosts` INT          NOT NULL,
    PRIMARY KEY (`idComments`),
    INDEX `fk_Comments_Posts1_idx` (`Posts_idPosts` ASC) VISIBLE,
    CONSTRAINT `fk_Comments_Posts1`
        FOREIGN KEY (`Posts_idPosts`)
            REFERENCES `lab6a`.`Posts` (`idPosts`)
            ON DELETE CASCADE
            ON UPDATE NO ACTION
)
    ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `lab6a`.`Posts_has_Categories`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `lab6a`.`Posts_has_Categories`
(
    `Posts_idPosts`           INT NOT NULL,
    `Categories_idCategories` INT NOT NULL,
    PRIMARY KEY (`Posts_idPosts`, `Categories_idCategories`),
    INDEX `fk_Posts_has_Categories_Categories1_idx`
(`Categories_idCategories` ASC) VISIBLE,
    INDEX `fk_Posts_has_Categories_Posts1_idx` (`Posts_idPosts` ASC)
VISIBLE,
    CONSTRAINT `fk_Posts_has_Categories_Posts1`
```

```
        FOREIGN KEY (`Posts_idPosts`)
            REFERENCES `lab6a`.`Posts` (`idPosts`)
            ON DELETE CASCADE
            ON UPDATE NO ACTION,
    CONSTRAINT `fk_Posts_has_Categories_Categories1`
        FOREIGN KEY (`Categories_idCategories`)
            REFERENCES `lab6a`.`Categories` (`idCategories`)
            ON DELETE CASCADE
            ON UPDATE NO ACTION
)
    ENGINE = InnoDB;



# 2 Skapa procedures för att lägga till innehåll
/* Skapa de procedurer som behövs för att lägga till innehåll ange
lämpliga parametrar. Du ska implementera procedurer för att:


    Lägga till en skribent. Ska kunna köras som t ex CALL
add_author("Ada", "Ada Lovelace");
    Lagga till en kategori
    Lägga till en bloggpost med rubrik, datum och innehåll för en viss
skribent
    Koppla ihop en bloggpost med en kategori
    Lägga till en kommentar till ett bloggpost
*/

drop procedure if exists add_author;
delimiter //

create procedure add_author(in name_param varchar(32), in username_param
varchar(16))
begin

    insert into Authors(name, username)
    values (name_param, username_param);

end //
delimiter ;

-- adds 5 authors
call add_author('Mia', 'Minator250'); -- a1
CALL add_author('Ada', 'Ada Lovelace'); -- a2
call add_author('Benjamin Appelberg', 'bwauu'); -- a3
call add_author('Martin Goblirsch', 'Mat-GO'); -- a4
call add_author('Jakob Baldin', 'jb artist');
-- a5

-- see 5 Authors in table
select *
from Authors;

drop procedure if exists add_category;
```

```
delimiter //

create procedure add_category(in name_param varchar(32))
begin

    insert into Categories(name)
    values (name_param);

end //
delimiter ;

-- adds 10 categories
call add_category('Technology'); -- Kategorinamn för id = 1
call add_category('Philosophy'); -- Kategorinamn för id = 2
call add_category('Skateboarding'); -- Kategorinamn för id = 3
call add_category('Social'); -- Kategorinamn för id = 4
call add_category('Sports'); -- Kategorinamn för id = 5
call add_category('Shopping'); -- Kategorinamn för id = 6
call add_category('Traveling'); -- Kategorinamn för id = 7
call add_category('Music'); -- Kategorinamn för id = 8
call add_category('Physics'); -- Kategorinamn för id = 9
call add_category('Gaming');
-- Kategorinamn för id = 10

-- see 10 Categories from Categories table
select *
from Categories;


drop procedure if exists add_post;
delimiter //

create procedure add_post(in content_param varchar(64), in param_subject
varchar(32), in param_post_time datetime,
                          in param_idAuthors int)

begin
    insert into Posts(content, subject, posted, Authors_idAuthors)
    values (content_param, param_subject, param_post_time,
param_idAuthors);

end //
delimiter ;

call add_post('The byte is a unit of dinformation that consists of eight
bits.', 'Data Science is cool',
              '2020-09-14 14:18:17', 1); -- p1
call add_post('I think therefore I am', 'My favorite quote', '2021-09-14
23:18:17', 2); -- p2
call add_post('IsmanmerelyamistakeofGods?OrGodmerelyamistakeofmans?',
'Cool Friedrich Nietzsche', '2021-12-01 12:18:17',3); -- p3
call add_post('God is dead! He remains dead! And we have killed him.', 'My
favorite quote', '2021-12-03 13:18:17',3); -- p4
```

```
call add_post('Happiness is the highest good', 'Quote by Aristotle',
'2021-12-12 13:18:23', 3); -- p5
call add_post('Man is condemned to be free', 'Quote by Jean-Paul Sartre',
'2021-12-13 23:18:17', 3); -- p6
call add_post('Finns det någon man kan chatta med här?', 'Chattkompis?',
'2021-12-15 22:12:17', 3); -- p7
call add_post('The unexamined life is not worth living', 'Socrates most
famous quote ', '2021-12-20 15:12:17', 3); -- p8
call add_post('I´ve collected data from runners one time before.',
'Databases is fun!', '2022-01-12 22:12:17', 5); -- p9
call add_post('I should tell my students more about my working
experience', 'To consider', '2022-01-28 18:12:17',5); -- p10


drop procedure if exists add_comment_TO_post;
delimiter //

create procedure add_comment_TO_post(in param_name varchar(64),
param_subject varchar(32), in param_content varchar(64), in param_postid
int)
begin

    insert into Comments(name, subject, content, posted, Posts_idPosts)
    values (param_name, param_subject, param_content, now(),
param_postid);

end //
delimiter ;

-- adds 11 comments to given PostId.
call add_comment_TO_post('Stacy', 'Well that is true, but!', 'I think you
meant "Digital Information" not dinformation. Noob!', 1); -- c1
call add_comment_TO_post('philosophybasics', 'PLAGIAT!!!','That is not a
quote of your own!! It´s from René Descartes.', 2); -- c2
call add_comment_TO_post('Frank', 'Totally agree', 'Yeah I totally agree,
it would be a cooler learning experience ',10); -- c3
call add_comment_TO_post('Nickrad', 'Det var jag', 'Hallå jag försökte
ringa dig men du svarade inte ', 3); -- c4
call add_comment_TO_post('Nickrad', 'Jatack!', 'Jag skulle vilja chatta
med dig! :)', 7); -- c5
call add_comment_TO_post('Markus', 'Fredrich is da bomb!', 'I Love
Fredich! It´s my favorite quote too! ', 4); -- c6
call add_comment_TO_post('Ragnell', 'To Happiness', 'Happiness is Love,
Happiness is Life!', 5); -- c7
call add_comment_TO_post('Robot', 'I don´t know...', 'I am confused, where
Am I? Who Am I?', 5); -- c8
call add_comment_TO_post('My', 'Question', 'Could a Universe be a
boolean?', 1); -- c9
call add_comment_TO_post('SQL', 'Hoppas på svar!', 'Jaa vi kan chatta tror
jag.', 7); -- c10
call add_comment_TO_post('rand()', 'I generate random', 'Hey this is not
Java! >:(', 7); -- c11

# 3 Skapa innehåll
```

```
/* Fyll på med innehåll genom att anropa dina procedurer ovan.
   Det skan vara minst fem skribenter med från noll till fem bloggposter
var och totalt vara minst tio blogginlägg.
   Det ska även finnas totalt minst tio kommentarer och tio kategorier som
är kopplade till bloggposterna
   (alltså inte 10 till varje bloggpost). Redovisa alla queries som behövs
för att fylla databasen med innehåll.
   För texter och rubriker för inlägg, texter och rubriker samt
kommentarer använd någon form av lorem ipsumLänkar till en externa sida.
   för indata. Helt slumpade tecken eller likt ¨
   "aaaaaaaaa a a aaaaaa xxxxx x x xxxxxxxxxx x xx x xxxxxx x xxxx sss xs
Xa d a DA d aD adaDadad
   fegfwfgwe geewge eqffqfewefiweofnweoifnf wef ewfwef fewefwefewfew
fwefwieooewifoewioeiwjsdvsd. gsdgsdgrwgw eg we g ds gds dg ew egw gwgew"
är inte ok.
   */

insert into Posts_has_Categories(Posts_idPosts, Categories_idCategories)
values (1, 1),
       (2, 2),
       (3, 3),
       (4, 4),
       (5, 4),
       (6, 2),
       (7, 4),
       (7, 2),
       (8, 5),
       (9, 10),
       (10, 1);

-- all comment result with Posts table
select *
from Posts
         join Comments
              on idPosts = Posts_idPosts;

-- Categories with posts subject
select Categories.name, Posts.subject
from categories
         join Posts_has_Categories on
Posts_has_Categories.Categories_idCategories = Categories.idCategories
         join Posts on Posts_has_Categories.Posts_idPosts = Posts.idPosts;

-- all authors
select *
from Authors;

# 4 Ta bort kategori
/*
Skapa en procedure som när den anropas med namn på en kategori tar bort
kategorin.
*/
```

```
drop procedure if exists delete_category;
delimiter //

create procedure delete_category(in name_param varchar(16))
begin

    delete
    from Categories
    where Categories.name = name_param;


end //
delimiter ;

-- q4 result
select *
from categories
where Categories.name = 'Skateboarding';

call delete_category('Skateboarding');


select *
from Posts
         left outer join Posts_has_Categories
                         on Posts.idPosts = Posts_idPosts
         left outer join categories
                         on categories.idCategories =
categories_idCategories;

select *
from Posts_has_Categories;

# 5 Visa alla kommentarer
/*
    Skapa en procedur som visar alla kommenterar för ett givet
blogginlägg.
    Ska kunna köras med t ex CALL show_comments(23); där 23 är id för ett
blogginlägg.
*/


drop procedure if exists show_comments_by_idPosts;
delimiter //

create procedure show_comments_by_idPosts(in param_Posts_idPosts int)
begin

    select *
    from Comments
        join Posts on Comments.Posts_idPosts = Posts.idPosts
    where Comments.Posts_idPosts = param_Posts_idPosts;

end //
```

```
delimiter ;

call show_comments_by_idPosts(7);

# 6 Visa aktivitetsnivå
/* Skriv en procedure som visar en kolumn med namn på skribenter och
en kolumn för activity_level är "low" för de med 1-2 inlägg,
 "medium" för de med 3-5 inlägg och "high" för de med fler än 5 inlägg.
 Den behöver ej kunna hantera de skribenter som skrivit 0 inlägg
 (men om du vill kan de som skrivit 0 inlägg få activity_level satt till
"not active").
 */


drop procedure if exists authors_activity;
delimiter //

create procedure authors_activity()
begin

    select Authors.username,
           case
               when count(Posts.idPosts) < 3
                   then 'low'
               when count(Posts.idPosts) < 6
                   then 'medium'

               else 'high'
               end
    from Authors
             join Posts on Authors.idAuthors = Posts.Authors_idAuthors
    group by Authors.username;

end //
delimiter ;

call authors_activity();

# 7 no_of_comments(post_id)
-- Skapa en funktion som returnerar hur många kommentarer som är kopplade
till ett blogginlägg.

drop function if exists no_of_comments;

drop function if exists no_of_comments;
delimiter //
create function no_of_comments(param_post_id int)
    returns int
    deterministic

    begin
    declare total_comments int;
    set total_comments = (
        select count(*)
```

```
        from Posts
                join Comments on Posts.idPosts = Comments.Posts_idPosts
        where idPosts = param_post_id);

    return total_comments;

end //
delimiter ;

select *
from Comments
where Posts_idPosts = 6;
-- no comments yet
-- adding 2 comments to post with empty commentfield
call add_comment_TO_post('Eminem', 'I like to rap', 'Hello my name is Slim
Shady!', 6); -- c12
call add_comment_TO_post('Cristian Ronaldo', 'Soccer is love', 'I play
fotball and I score every game', 6); -- c12
select no_of_comments(6);

# 8 no_of_categories(post_id)
-- Skapa en funktion som returnerar hur många kategorier som är kopplade
till ett blogginlägg.


drop function if exists no_of_categories;
delimiter //
create function no_of_categories(param_post_id int)
    returns int
    deterministic
begin
    declare total_comments int;
    set total_comments = (
        select count(*)
        from categories
                join Posts_has_Categories on
Posts_has_Categories.Categories_idCategories = idCategories
                join Posts on Posts_has_Categories.Posts_idPosts =
idPosts
        where Posts.idPosts = param_post_id
    );

    return total_comments;

end //
delimiter ;

select no_of_categories(7);

# 9 time_since_posting(datetime)
/*
Skapa en funktion som tar ett datetime och returnerar en textsträng som
talar om hur länge sedan givet datetime var.
```

```
För inlägg som är skapade inom 1 h från när funktionen körs ska den skriva
ut "new"
annars ska den skriva ut "less than 1 day" om det gått mindre än en dag
och
för övriga ska det skrivas ut hur många dagar som gått sedan som inlägget
skrevs ex , "1 day", "2 days", "132 days".
Vill du göra en annan uträkning med tid utifrån andra alternativ är det ok
(så länge det är samma eller högre komplexitet).

*/

call add_post('Everyone is just typing random stuff in different
categories', 'Not confused at all #new', now(),4); -- p11


drop function if exists time_since_posting;
delimiter //

create function time_since_posting(param_datetime datetime)
    returns varchar(64)
    deterministic
begin
    declare message varchar(64);
    declare days_since int;
    declare hours_since int;

    set days_since = (timestampdiff(day, param_datetime, now()));
    set hours_since = (timestampdiff(hour, param_datetime, now()));

    if hours_since < 1 then
        set message = 'new';

    elseif days_since < 1 then
        set message = 'less than one day';

    else
        set message = concat(days_since, ' days old');

    end if;

    return message;

end //
delimiter ;


select subject, time_since_posting(posted)
from posts;


# 10 all_posts
/* Skapa en vy all_posts som är lista med alla blogginlägg med kolumner
för namn på bloggare,
```

```
 id för inlägg, rubrik samt använd även funktionen time_since_posting för
att visa hur gamla inlägg är.
 Visa även kolumner med värden för hur många kommentarer repektive hur
många kategorier som är kopplade till varje blogginlägg. Sortera på
tid+datum för när de skapades, nyast överst.
*/
create or replace view all_posts as
select Authors.name,
       Posts.idPosts,
       Posts.subject,
       time_since_posting(Posts.posted),
       no_of_comments(Posts.idPosts),
       no_of_categories(Posts.idPosts)
from Authors
         join Posts on Posts.Authors_idAuthors = Authors.idAuthors
order by Posts.posted desc;

select *
from all_posts;

# 11 Egen procedure eller function
/* Skapa en egen procedure eller function som gör något som är
meningsfullt och användbart med databasen.
Hitta på något eget som kan passa och vara intressant. Skriv tydliga
kommentarer och queries som visar hur den används och fungerar.*/

/*
SP edit_post takes two arguments param_idPost INT & param_content
VARCHAR(255) and allows
authors to edit their own posted. when edit_post is called and content is
set, the last_edit will be updated
to the time in which time edit_post was invoked.
*/
drop procedure if exists edit_post;
delimiter //

create procedure edit_post(in param_idPost int, in param_content
varchar(255))
begin

    update Posts
    set content   = param_content,
        last_edit = now()
    where idPosts = param_idPost;

end //
delimiter ;

select *
from Posts
where idPosts = 1;

call edit_post(1, 'The byte is a unit of digital information that consists
of eight bits.');
```

```
select *
from Posts
where idPosts = 1;


# 12 Egen trigger
/*  Skapa en egen trigger som gör något som är meningsfullt och användbart
med databasen.
    Hitta på något eget som kan passa och vara intressant.
    Skriv tydliga kommentarer och queries som visar hur den används och
fungerar.
*/

/* Trigger censors to all insert & update actions on table Comments column
'content' containing words like 'jävlar" && 'shit'. If these words are
used in content column
   a errormessage will be displayed*/
drop trigger if exists censor_comment_insert;
delimiter //
create trigger censor_comment_insert
    before insert
    on Comments
    for each row
begin

    if new.content like '%jävlar%' OR new.content like '%shit%' then
        signal sqlstate '45000'
            set message_text = 'No curse words!';
    end if;
end //
delimiter ;

drop trigger if exists censor_comment_update;
delimiter //
create trigger censor_comment_update
    before update
    on Comments
    for each row
begin

    if new.content like '%jävlar%' OR new.content like '%shit%' then
        signal sqlstate '45000'
            set message_text = 'No curse words!';
    end if;
end //
delimiter ;

select *
from comments;

insert into Comments(name, subject, content, posted, Posts_idPosts)
values ('Troll', 'troll.com', 'JÄVLAR!', now(), 2);
```

```
insert into Comments(name, subject, content, posted, Posts_idPosts)
values ('TrollTryAgain', 'trollx.com', 'SHIT!', now(), 2);

update Comments
set content = 'Shit!'
where Posts_idPosts = 3;

select *
from comments;
```