```
USE lab3;

# Veckodagar
# a) Skriv en query som visar vilken veckodag det kommer vara på den sista dagen
nuvarande månad. (Använd now() för att avgöra vilken månad det är. Queryn ska
fungera för alla dagar hela året, alla år.)
SELECT DAYNAME(LAST_DAY(NOW()));

# b) Skriv en query som visar vilken veckodag det kommer vara på den första dagen
nästa månad. (Använd now() för att avgöra vilken nästa månad det är. Queryn ska
fungera för alla dagar hela året, alla år.)
SELECT DAYNAME(LAST_DAY(NOW()) + INTERVAL 1 DAY);


# Dagar sen terminsstart HT-20
# Skriv en query som visar hur många dagar det gått sen 2020-08-31 (terminsstarten
HT-20 vid UU.)
SELECT TIMESTAMPDIFF(DAY, '2020-08-31', NOW());


# midnight_countdown()
# Skriv en procedure som när den anropas visar hur många minuter det är kvar till
midnatt.
DROP PROCEDURE IF EXISTS midnight_countdown;
DELIMITER //
CREATE PROCEDURE midnight_countdown()
      SELECT TIMESTAMPDIFF(MINUTE, NOW(), DATE(NOW()) + INTERVAL 1 DAY) AS
minutes_until_midnight;
DELIMITER ;

CALL midnight_countdown();


# born_in_february
# Skapa en vy som visar alla users som är födda i februari.
DROP VIEW IF EXISTS born_list;
CREATE VIEW born_list AS
      SELECT users.username, users.birthdate
      FROM users
      WHERE MONTH(birthdate) = 2;

SELECT * FROM born_list;


# happy_birthday(month, day)
# Skriv en procedure som visar förnamn och efternamn på alla users som har
födelsedag på given månad och dag.
DROP PROCEDURE IF EXISTS happy_birthday;
DELIMITER //
CREATE PROCEDURE happy_birthday(month INT, day INT)
BEGIN
      SELECT first_name, last_name
      FROM Users
      WHERE month(birthdate) = month AND dayofmonth(birthdate) = day;
END //
DELIMITER ;

CALL happy_birthday(6, 17);
```

```
# age(birthdate)
# För users gör en funktion som räknar ut ålder (i år) utifrån ett födelsedatum.
# Ska kunna användas med t ex SELECT first_name, last_name, birthdate,
age(birthdate) FROM users ORDER BY birthdate DESC;
DELIMITER //
DROP FUNCTION IF EXISTS age;
CREATE FUNCTION age(birthdate DATE)
RETURNS INT
NO SQL
      RETURN TIMESTAMPDIFF(YEAR, birthdate, NOW());
DELIMITER ;

SELECT first_name, last_name, birthdate, age(birthdate) FROM users ORDER BY
birthdate DESC;


# time_trigger
# För users skriv triggers så att det inte går att lägga in och inte ändra till
födelsedatum som ligger framåt i tiden.
DROP PROCEDURE IF EXISTS validate_birthdate;
DROP TRIGGER IF EXISTS time_trigger_insert;
DROP TRIGGER IF EXISTS time_trigger_update;

DELIMITER //
CREATE PROCEDURE validate_birthdate(IN birthdate DATE)
BEGIN
      IF TIMESTAMPDIFF(day, birthdate, DATE(NOW())) < 0 THEN
          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Kan inte vara född i
framtiden';
      END IF;
END //
DELIMITER ;

CREATE TRIGGER time_trigger_insert
BEFORE INSERT ON Users FOR EACH ROW CALL validate_birthdate(NEW.birthdate);

CREATE TRIGGER time_trigger_update
BEFORE UPDATE ON Users FOR EACH ROW CALL validate_birthdate(NEW.birthdate);

# Ändra till ett datum i framtiden för att se felmeddelande, t.ex. 2030-01-01
UPDATE Users SET birthdate = '2000-01-01' WHERE id = 1;
SELECT birthdate FROM Users WHERE id = 1;


# orders_sent_but_not_recieved
# För orders skapa en vy som visar order-id för alla orders som är skickade men
inte mottagna av kund.
# I vyn ska finnas en kolumn, time_in_transit, som visar hur många timmar och
minuter som gått sen ordern skickades.
DROP VIEW IF EXISTS orders_sent_but_not_recieved;
CREATE VIEW orders_sent_but_not_recieved AS
    SELECT id, SEC_TO_TIME(TIMESTAMPDIFF(SECOND, sent_from_store, now())) AS
time_in_transit
    FROM Orders
      WHERE
            sent_from_store IS NOT NULL AND
            arrived_at_customer IS NULL;
```

```
SELECT * FROM orders_sent_but_not_recieved;


# fastest_delivery
# För orders skapa en vy som visar de 5 orders som det var kortast tid mellan att
de skickades tills att de kom fram till kund.
# Sortera på tid i stigande ordning (kortast tid överst).
DROP VIEW IF EXISTS fastest_delivery;
CREATE VIEW fastest_delivery AS
    SELECT id, SEC_TO_TIME(TIMESTAMPDIFF(SECOND, sent_from_store,
arrived_at_customer)) AS delivery_time
    FROM Orders
      WHERE
            sent_from_store IS NOT NULL AND
            arrived_at_customer IS NOT NULL
      ORDER BY delivery_time
      LIMIT 5;

SELECT * FROM fastest_delivery;


# order_status()
# För orders skapa en procedure som visar order-id, och en kolumn för "Status" där
det står:
#    "Processing" för de orders som är mottagna men inte är skickade
#    "Sent" för de som är skickade men ej kommit fram till kund där det inte gått
mer än 14 dagar
#    "Lost?" för de som är skickade men ej kommit fram till kund och där det gått
mer än 14 dagar
#    "Fast" för de som kommit fram till kund inom 120 h
#    "Slow" för de som kommit fram med mer än 120 h i frakttid.
DROP PROCEDURE IF EXISTS order_status;

DELIMITER //
CREATE PROCEDURE order_status()
BEGIN
      SELECT id, received_at_store, sent_from_store, arrived_at_customer,
    CASE
        # "Processing" för de orders som är mottagna men inte är skickade
        WHEN
            received_at_store IS NOT NULL AND
            sent_from_store IS NULL
        THEN 'Processing'

        # "Sent" för de som är skickade men ej kommit fram till kund där det inte
gått mer än 14 dagar
        WHEN
                  sent_from_store IS NOT NULL AND
                  arrived_at_customer IS NULL AND
                  TIMESTAMPDIFF(DAY, DATE(sent_from_store), DATE(NOW())) <= 14
            THEN 'Sent'

        # "Lost?" för de som är skickade men ej kommit fram till kund och där det
gått mer än 14 dagar
        WHEN
                  sent_from_store IS NOT NULL AND
                  arrived_at_customer IS NULL AND
                  TIMESTAMPDIFF(DAY, DATE(sent_from_store), DATE(NOW())) > 14
            THEN 'Lost?'
```

```
        # "Fast" för de som kommit fram till kund inom 120 h
        WHEN
                  arrived_at_customer IS NOT NULL AND
                  TIMESTAMPDIFF(HOUR, sent_from_store, NOW()) < 120
            THEN 'Fast'

        # "Slow" för de som kommit fram med mer än 120 h i frakttid.
        WHEN
                  arrived_at_customer IS NOT NULL AND
                  TIMESTAMPDIFF(HOUR, sent_from_store, NOW()) >= 120
            THEN 'Slow'
      END
      FROM Orders;
END //
DELIMITER ;

CALL order_status();


# egen trigger
# Skapa en egen trigger som gör något som är meningsfullt och användbart med
databasen (orders eller users).
# Hitta på något eget som kan passa och vara intressant.
# Skriv tydliga kommentarer och queries som visar hur den används och fungerar.
# Den ska hantera något som har med tid eller datum att göra.
DROP TRIGGER IF EXISTS arrival_time_trigger;

DELIMITER //
# En trigger som kollar så att paket inte kan få ett leveransdatum som ligger innan
utskickningsdatum
CREATE TRIGGER arrival_time_trigger
BEFORE UPDATE ON Orders
      FOR EACH ROW
        IF (TIMESTAMPDIFF(SECOND, NEW.arrived_at_customer, NEW.sent_from_store) >
0) THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Paketet kan inte komma fram
innan det har skickats';
        END IF //
DELIMITER ;

UPDATE Orders SET sent_from_store = '2022-01-01' WHERE id = 10;
# Ändra till ett datum före 2022-01-01 för att se felmeddelande
UPDATE Orders SET arrived_at_customer = '2023-01-01' WHERE id = 10;

# SELECT * FROM Orders WHERE id = 10;


# egen procedure eller function
# Skapa en egen procedure eller function som gör något som är meningsfullt och
användbart med databasen (orders eller users).
# Hitta på något eget som kan passa och vara intressant.
# Skriv tydliga kommentarer och queries som visar hur den används och fungerar.
# Den ska hantera något som har med tid eller datum att göra.
DROP PROCEDURE IF EXISTS birthdays_next_week;
DELIMITER //
# Ta fram en lista av alla användare som fyller år inom en vecka (så man inte
glömmer att säga grattis eller köpa en liten present)
CREATE PROCEDURE birthdays_next_week()
```

```
      SELECT
            first_name,
            last_name,
        DAYNAME(MAKEDATE(YEAR(NOW()), DAYOFYEAR(birthdate))) AS birthday,
            MAKEDATE(YEAR(NOW()), DAYOFYEAR(birthdate)) AS birthday_date
      FROM Users
      WHERE
            DAYOFYEAR(birthdate) - DAYOFYEAR(NOW()) > 0 AND
            DAYOFYEAR(birthdate) - DAYOFYEAR(NOW()) < 7
      ORDER BY birthday_date;
DELIMITER ;

CALL birthdays_next_week();
```