ДОДАТОК Г

SQL-код бази даних у PostgreSQL

```
-- CREATE DATABASE Tickets_booking;

-- DROP SCHEMA public CASCADE;
-- CREATE SCHEMA public;

-- Створення таблиць
-- -----------------------------------------------------
-- Table Type_transportations
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Type_transportations (
  transportation_ID SERIAL NOT NULL,
  type_name TEXT NOT NULL,
  PRIMARY KEY (transportation_ID));


-- -----------------------------------------------------
-- Table Firms
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Firms (
  firm_ID SERIAL NOT NULL,
  idTransportation INT NOT NULL,
  name TEXT NOT NULL,
  email TEXT NOT NULL,
  tel TEXT NOT NULL,
  region_activity TEXT NULL,
  PRIMARY KEY (firm_ID),
  FOREIGN KEY (idTransportation) REFERENCES Type_transportations
(transportation_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);


-- -----------------------------------------------------
-- Table Transport
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Transport (
  transport_ID SERIAL NOT NULL,
  name TEXT NOT NULL,
  number TEXT NOT NULL,
  countplace INT NOT NULL,
  PRIMARY KEY (transport_ID));


CREATE TABLE Del_transport (LIKE Transport INCLUDING ALL);

ALTER TABLE Del_transport ADD COLUMN time TIMESTAMP NULL;


-- -----------------------------------------------------
-- Table Flights
-- -----------------------------------------------------
CREATE TYPE FLIGHTSTATUS AS ENUM('Скасований', 'Відбувся', 'Очікується');

CREATE TABLE IF NOT EXISTS Flights (
  flight_ID SERIAL NOT NULL,
  idFirm INT NOT NULL,
```

```
  idTransport INT NOT NULL,
  f_from TEXT NULL,
  f_to TEXT NULL,
  f_distance FLOAT NULL,
  starttime TIMESTAMP NOT NULL,
  endtime TIMESTAMP NULL,
  status FLIGHTSTATUS DEFAULT 'Очікується',
  PRIMARY KEY (flight_ID),
  FOREIGN KEY (idFirm) REFERENCES Firms(firm_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idTransport) REFERENCES Transport(transport_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

-- -----------------------------------------------------
-- Table Customers
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Customers (
  customer_ID SERIAL NOT NULL,
  c_name TEXT NOT NULL,
  c_birthday DATE NOT NULL,
  PRIMARY KEY (customer_ID));


-- -----------------------------------------------------
-- Table Type_tickets
-- -----------------------------------------------------
CREATE TYPE TYPETICKET AS ENUM('Купе дорослий', 'Купе дитячий', 'Купе
студентський', 'Плацкарт дорослий',
      'Плацкарт дитячий', 'Плацкарт студентський', 'Бізнес клас', 'Економ
клас', 'Перший клас', 'Автобусний',
      'Автобусний студентський', 'Морський', 'Морський VIP', 'Морський
економ');

CREATE TABLE IF NOT EXISTS Type_tickets(
   typeticket_ID SERIAL NOT NULL,
   type_afterpay TYPETICKET NOT NULL,
   percent_afterpay INT NOT NULL,
   PRIMARY KEY (typeticket_ID));


-- -----------------------------------------------------
-- Table Cities
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Cities (
  city_ID SERIAL NOT NULL,
  city_name TEXT NOT NULL,
  country TEXT NOT NULL,
  PRIMARY KEY (city_ID));


-- -----------------------------------------------------
-- Table Flights_has_Cities
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Flights_has_Cities (
  primaryID SERIAL NOT NULL,
  idFlight INT NOT NULL,
  idCity1 INT NOT NULL,
  idCity2 INT NULL,
  PRIMARY KEY (primaryID),
  FOREIGN KEY (idFlight) REFERENCES Flights (flight_ID)
  ON DELETE CASCADE
```

```
  ON UPDATE CASCADE,
  FOREIGN KEY (idCity1) REFERENCES Cities (city_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idCity2) REFERENCES Cities (city_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

-- -----------------------------------------------------
-- Table Airplane_tickets
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Airplane_tickets (
  a_ticket_ID SERIAL NOT NULL,
  at_price DECIMAL NULL,
  idType_tickets INT NOT NULL,
  Flights_has_Cities_primaryID INT NOT NULL,
  PRIMARY KEY (a_ticket_ID),
  FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES Flights_has_Cities
(primaryID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);


-- -----------------------------------------------------
-- Table Routes
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Routes (
  route_ID SERIAL NOT NULL,
  Flights_has_Cities_primaryID INT NOT NULL,
  idCity INT NOT NULL,
  PRIMARY KEY (route_ID),
  FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES Flights_has_Cities
(primaryID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idCity) REFERENCES Cities (city_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

-- -----------------------------------------------------
-- Table Train_tickets
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Train_tickets (
  t_ticket_ID SERIAL NOT NULL,
  tt_price DECIMAL NULL,
  idType_tickets INT NOT NULL,
  idRoute INT NOT NULL,
  PRIMARY KEY (t_ticket_ID),
  FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idRoute) REFERENCES Routes (route_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);


-- -----------------------------------------------------
-- Table Bus_tickets
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Bus_tickets (
```

```
  b_ticket_ID SERIAL NOT NULL,
  bt_price DECIMAL NULL,
  idType_tickets INT NOT NULL,
  idRoute INT NOT NULL,
  PRIMARY KEY (b_ticket_ID),
  FOREIGN KEY (idRoute) REFERENCES Routes (route_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);


-- -----------------------------------------------------
-- Table Ship_tickets
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Ship_tickets (
  s_ticket_ID SERIAL NOT NULL,
  st_price DECIMAL NULL,
  idType_tickets INT NOT NULL,
  Flights_has_Cities_primaryID INT NOT NULL,
  PRIMARY KEY (s_ticket_ID),
  FOREIGN KEY (idType_tickets) REFERENCES Type_tickets (typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Flights_has_Cities_primaryID) REFERENCES Flights_has_Cities
(primaryID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);


-- -----------------------------------------------------
-- Table Train_tickets_has_Customers
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Train_tickets_has_Customers (
  Train_tickets_t_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  t_booking_time TIMESTAMP NOT NULL,
  t_numwagon INT NOT NULL,
  t_seat INT NOT NULL,
  comments TEXT NULL,
  PRIMARY KEY (Train_tickets_t_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Train_tickets_t_ticket_ID) REFERENCES Train_tickets
(t_ticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers (customer_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);


-- -----------------------------------------------------
-- Table Airplane_tickets_has_Customers
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Airplane_tickets_has_Customers (
  Airplane_tickets_a_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  a_booking_time TIMESTAMP NOT NULL,
  at_class TEXT NULL,
  a_seat INT NOT NULL,
  comments TEXT NULL,
  PRIMARY KEY (Airplane_tickets_a_ticket_ID, Customers_customer_ID),
```

```
  FOREIGN KEY (Airplane_tickets_a_ticket_ID) REFERENCES Airplane_tickets
(a_ticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers (customer_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

-- -----------------------------------------------------
-- Table Bus_tickets_has_Customers
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Bus_tickets_has_Customers (
  Bus_tickets_b_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  b_booking_time TIMESTAMP NOT NULL,
  b_seat INT NOT NULL,
  comments TEXT NULL,
  PRIMARY KEY (Bus_tickets_b_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Bus_tickets_b_ticket_ID) REFERENCES Bus_tickets (b_ticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers (customer_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);


-- -----------------------------------------------------
-- Table Ship_tickets_has_Customers
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Ship_tickets_has_Customers (
  Ship_tickets_s_ticket_ID INT NOT NULL,
  Customers_customer_ID INT NOT NULL,
  s_booking_time TIMESTAMP NOT NULL,
  s_class TEXT NULL,
  s_seat INT NOT NULL,
  comments TEXT NULL,
  PRIMARY KEY (Ship_tickets_s_ticket_ID, Customers_customer_ID),
  FOREIGN KEY (Ship_tickets_s_ticket_ID) REFERENCES Ship_tickets
(s_ticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Customers_customer_ID) REFERENCES Customers (customer_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

-- -----------------------------------------------------
-- Table Transport_has_Type_tickets
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS Transport_has_Type_tickets (
  Transport_idTransport INT NOT NULL,
  Type_ticketa_idType_tickets INT NOT NULL,
  count_seats_type INT NOT NULL,
  PRIMARY KEY (Transport_idTransport, Type_ticketa_idType_tickets),
  FOREIGN KEY (Transport_idTransport) REFERENCES Transport (transport_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Type_ticketa_idType_tickets) REFERENCES Type_tickets
(typeticket_ID)
  ON DELETE CASCADE
  ON UPDATE CASCADE);

-- -----------------------------------------------------
-- ЗАПОВНЕННЯ ТАБЛИЦЬ
```

```
-- ----------------------------------------------------
BEGIN ;
INSERT INTO Type_transportations(type_name)
VALUES ('Повітряне'),('Колійне'),('Водне'),('Наземне');
COMMIT ;

-- SELECT * FROM Type_transportations;

INSERT INTO Firms(idTransportation, name, email, tel, region_activity)
VALUES (1,'МАУ','uia@flyuia.com','+38 (044) 581-50-50','Україна'),
       (1,'KLM','KLM.Ukraine@klm.nl','+31 (0) 20 649 91 23','Нідерланди'),
       (1,'Emirates Airlines','pr@emirates.com','600 555 555','ОАЕ'),
       (1,'British Airways','ba.ukraine@ba.com','+44 (20) 8738
5050','Англія'),
       (1,'Wizz Air','info@wizzair.com','+380 89 320 25 33','Венгрія'),
       (1,'Turkish Airlines','ievsales@thy.com','+90 444 0
8491','Туреччина'),
       (1,'Lufthansa','info@lufthansa.com','+49 (69) 696 46 01','Німеччина'),
       (2,'Укрзалізниця','cgk@uz.gov.ua','0-800-50-311',''),
       (2,'Polrail Service','office@polrail.com','+48 52 332 57 81',''),
       (2,'CFR Călători','presa@cfrcalatori.ro','0800.88.44.44',''),
       (2,'БЖД','ns@rw.by','(+375 17) 225 49 46','Білорусь'),
       (3,'Costa Cruises','customercare@us.costa.it','1-800-462-6782',''),
       (3,'MSC','UA539-odessa@msc.com','+380 48 784 7272',''),
       (3,'Royal Caribbean','cliente@rccl.com.br','(888) 724-7447',''),
       (3,'Celebrity Cruises','CelebrityEngagementCenter@celebrity.com','00 1
305-341-0205',''),
       (4,'AUTOLUX','help@autolux.ua','+38 044 594 95 00',''),
       (4,'Gunsel','info@gunsel.com.ua','+38 (044) 525-45-05',''),
       (4,'EuroClub','euroclubbus@gmail.com','+38(044)486 79 00',''),
       (4,'Orionbus','rionbilet@gmail.com','050 010 0104',''),
       (4,'Ecolines','help@ecolines.ua','+38 044 594 90 10',''),
       (4,'EAST WEST EUROLINES','support@ewe.ua','+380988154444',''),
       (4,'TransTempo','transtempo@ukr.net','+38 (067) 467-44-77','');

-- SELECT * FROM Firms;

INSERT INTO Customers(c_name, c_birthday)
VALUES ('Тарасович Ратимир Вадимович', '1952-08-10'),
       ('Ніколенко Ромашка Зорянівна', '1963-05-17'),
       ('Кириленко Муховіст Полянович', '1987-03-19'),
       ('Алиськевич Мар''ян Устимович', '1981-07-27'),
       ('Савка Щедра Фролівна', '1976-05-04'),
       ('Вітренко Власт Денисович', '1995-11-24'),
       ('Смолій Ярослав Зорянович', '1997-09-11'),
       ('Міняйло Йосифата Ярославівна', '1979-11-24'),
       ('Семків Йоган Артемович', '1991-02-05'),
       ('Нечитайло Данко Мстиславович', '2000-08-14'),
       ('Сомко Яромира Остапівна', '1994-09-17'),
       ('Токарчук Родіон Найденович', '1954-12-21'),
       ('Соломченко Хотимир Златович', '1996-08-09'),
       ('Лупійчук Дан Жданович', '1982-07-25'),
       ('Лазорський Никифор Олександрович', '1978-06-01'),
       ('Довгань Гаїна Тимурівна', '1977-06-30'),
       ('Юденко Олесь Денисович', '1967-12-17'),
       ('Пастушенко Ніна Ростиславівна', '1969-03-26'),
       ('Пустовойт Колодар Богданович', '1972-05-30'),
       ('Боєчко Лев Антонович', '1963-10-26'),
       ('Гриневецький Хотян Сарматович', '1995-07-15'),
       ('Барвінський Антон Охримович', '1991-04-09'),
       ('Кущ Югина Арсенівна', '1958-10-21'),
       ('Замора Єгор Жданович', '1956-07-30'),
       ('Німчук Злотан Найденович', '1982-03-12'),
```

```
        ('Німченко Йосифата Борисівна', '1956-05-21'),
        ('Ганущак Фауст Антонович', '1958-10-09'),
        ('Смішко Іларіон Ігорович', '1979-10-30'),
        ('Лещинська Жозефіна Августинівна', '1951-12-05'),
        ('Медяник Йозеф Богуславович', '1990-03-04'),
        ( 'Тихий Тихон Юхимович', '1977-04-30'),
        ( 'Мулярчук Хвалимир Остапович', '1994-01-07'),
        ( 'Мисько Живосил Чеславович', '1951-03-11'),
        ( 'Дурдинець Триріг Любомирович', '1971-07-10'),
        ( 'Магура Ізяслав Азарович', '1972-03-22'),
        ( 'Вінтоняк Альберт Пилипович', '1980-01-11'),
        ( 'Чалий Йозеф Костянтинович', '1952-10-24'),
        ( 'Могиленко Йосип Сарматович', '1991-07-10'),
        ( 'Коструба Щастислав Федорович', '1996-08-08'),
        ( 'Павлович Гладко Пилипович', '1963-09-12'),
        ( 'Петрицький Зборислав Давидович', '1950-04-18'),
        ( 'Сімович Цвітан Августинович', '1957-07-29'),
        ( 'Мацелюх Євстафій Августинович', '1953-09-01'),
        ( 'Чубатий Івантослав Давидович', '1960-10-21'),
        ( 'Сорока Юрій Гордиславович', '1993-06-12'),
        ( 'Вайда Турбрід Драганович', '1991-09-28'),
        ( 'Мухопад Царук Адріанович', '1982-04-20'),
        ( 'Бурбан Вернислав Артемович', '1993-03-05'),
        ( 'Штинь Атрей Русланович', '1956-11-19'),
        ( 'Деркач Лютобор Романович', '1966-07-25');

-- SELECT * FROM Customers;"

INSERT INTO Type_tickets (type_afterpay, percent_afterpay)
VALUES ( 'Купе дорослий', 50),
       ( 'Купе студентський', 30),
       ( 'Купе дитячий', 10),
       ( 'Плацкарт дорослий', 0),
       ( 'Плацкарт дитячий', -30),
       ( 'Плацкарт студентський', -50),
       ( 'Бізнес клас', 150),
       ( 'Економ клас', -10),
       ( 'Перший клас', 50),
       ( 'Автобусний', 0),
       ( 'Автобусний студентський', -20),
       ( 'Морський', 0),
       ( 'Морський VIP', 75),
       ( 'Морський економ',-15);

-- SELECT * FROM Type_tickets;

INSERT INTO Cities (country, city_name)
VALUES ('Молдова', 'Кишинів'),
       ('Греція', 'Афіни'),
       ('Хорватія', 'Загреб'),
       ('Швеція', 'Стокгольм'),
       ('Португалія', 'Лісабон'),
       ('Словаччина', 'Братислава'),
       ('Італія', 'Рим'),
       ('Македонія', 'Скоп’є'),
       ('Андорра', 'Андорра-ла-Велья'),
       ('Бельгія', 'Брюссель'),
       ('Чорногорія', 'Подгориця'),
       ('Данія', 'Копенгаген'),
       ('Ліхтенштейн', 'Вадуц'),
       ('Люксембург', 'Люксембург'),
       ('Великобританія', 'Лондон'),
       ('Словенія', 'Любляна'),
```

```
        ('Мальта', 'Валлетта'),
        ('Сан-Марино', 'Сан-Марино'),
        ('Польща', 'Варшава'),
        ('Німеччина', 'Берлін'),
        ('Сербія', 'Белград'),
        ('Швейцарія', 'Берн'),
        ('Чехія', 'Прага'),
        ('Литва', 'Вільнюс'),
        ('Ірландія', 'Дублін'),
        ('Албанія', 'Тирана'),
        ('Болгарія', 'Софія'),
        ('Ісландія', 'Рейк’явік'),
        ('Франція', 'Париж'),
        ('Білорусь', 'Мінськ'),
        ('Боснія і Герцеговина', 'Сараєво'),
        ('Румунія', 'Бухарест'),
        ('Фінляндія', 'Гельсінкі'),
        ('Ватикан', 'Ватикан'),
        ('Норвегія', 'Осло'),
        ('Австрія', 'Відень'),
        ('Латвія', 'Рига'),
        ('Іспанія', 'Мадрид'),
        ('Угорщина', 'Будапешт'),
        ('Монако', 'Монако'),
        ('Естонія', 'Таллінн'),
        ('Нідерланди', 'Амстердам'),
        ('Україна','Вінниця'),
        ('Україна','Луцьк'),
        ('Україна','Дніпро'),
        ('Україна','Донецьк'),
        ('Україна','Житомир'),
        ('Україна','Ужгород'),
        ('Україна','Запоріжжя'),
        ('Україна','Івано-Франківськ'),
        ('Україна','Київ'),
        ('Україна','Кропивницький'),
        ('Україна','Львів'),
        ('Україна','Миколаїв'),
        ('Україна','Луганськ'),
        ('Україна','Одеса'),
        ('Україна','Полтава'),
        ('Україна','Рівне'),
        ('Україна','Суми'),
        ('Україна','Тернопіль'),
        ('Україна','Харків'),
        ('Україна','Херсон'),
        ('Україна','Хмельницький'),
        ('Україна','Черкаси'),
        ('Україна','Чернівці'),
        ('Україна','Чернігів');

-- SELECT * FROM Cities;

CREATE OR REPLACE FUNCTION insertInDel_Transport() RETURNS trigger AS
$saveDelTransport$
    BEGIN
        INSERT INTO Del_transport(transport_ID, name, number, countplace,
time)
        VALUES(OLD.transport_ID, OLD.name, OLD.number, OLD.countplace,
NOW());
        RETURN NULL;
    END;
$saveDelTransport$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER saveDelTransport BEFORE DELETE ON Transport
   FOR EACH ROW EXECUTE PROCEDURE insertInDel_Transport();


INSERT INTO Transport(name, number, countplace)
VALUES ( 'Boeing', '777', 250),
       ( 'Airbus', 'A340', 170),
       ( 'Airbus', 'A330', 125),
       ( 'Boeing', '747', 245),
       ( 'Boeing Next Generation', '737', 290),
       ( 'Boeing Classic', '737', 230),
       ( 'AN', '225', 115),
       ( 'AN', '13', 134),
       ( 'AN', '74', 100),
       ( 'AN', '275', 110),
       ( 'AN', '132G', 265),
       ( 'AN', '28', 330),
       ( 'AN', '178', 110),
       ( 'Douglas', 'DC-1', 285),
       ( 'Dassault Falcon', '900', 280),
       ( 'Farman Goliath', 'F.60', 270),
       ( 'Dassault Falcon', '7X', 260),
       ( 'Богдан', 'AE5501AA', 25),
       ( 'Neoplan', 'AE5740KP', 30),
       ( 'Neoplan', 'AX3116BM', 40),
       ( 'ЕТАЛОН', 'BM6175CM', 30),
       ( 'Богдан', 'BC3458MK', 25),
       ( 'ЕТАЛОН', 'BI7837EA', 28),
       ( 'Богдан', 'AB1994IA', 25),
       ( 'ISUZU', 'AE1095AA', 32),
       ( 'Богдан', 'AE1095AA', 25),
       ( 'ЕТАЛОН', 'BH4091AA', 28),
       ( 'ЕТАЛОН', 'BC4780MB', 28),
       ( 'ЕТАЛОН', 'BO5913CP', 28),
       ( 'Богдан', 'AE4858OX', 25),
       ( 'ISUZU', 'BC6097MK', 28),
       ( 'Neoplan', 'KA2376BT', 30),
       ( 'ISUZU', 'AT7929EP', 26),
       ( 'GULERYUZ', 'BK3137HH', 24),
       ( 'Богдан', 'AC1501EO', 25),
       ( 'ISUZU', 'AO2753EP', 29),
       ( 'ISUZU', 'KA1002OP', 34),
       ( 'Богдан', 'BT1552CC', 25),
       ( 'ISUZU', 'AB2775HT', 33),
       ( 'Neoplan', 'BI7798EX', 30),
       ( 'GULERYUZ', 'BC9664MB', 22),
       ( 'ЕТАЛОН', 'AM9254EO', 28),
       ( 'Neoplan', 'AT6426EP', 30),
       ( 'GULERYUZ', 'BX5339EM', 36),
       ( 'GULERYUZ', 'BI8403EE', 28),
       ( 'Neoplan', 'BH2137OH', 32),
       ( 'ЕТАЛОН', 'AA4243PT', 30),
       ( 'Neoplan', 'BC5474MK', 32),
       ( 'ISUZU', 'BM1049CM', 30),
       ( 'Neoplan', 'AX5109HX', 30),
       ( 'Південний експрес', '686/685', 686),
       ( 'Північний експрес','784/785', 841),
       ( 'Подільський експрес ','103/104', 670),
       ( 'Приазов`я', '989/988', 898),
       ( 'Промінь', '421/420', 309),
       ( 'Рось', '117/118', 602),
       ( 'Севастополець', '542/543', 634),
```

```
       ( 'Таврія', '835/834', 455),
       ( 'Хаджибей', '683/682', 643),
       ( 'Чайка', '093/094', 542),
       ( 'Enchantment of the Seas', '', 2405),
       ( 'Symphony of the Seas', '', 1300),
       ( 'Celebrity Xploration', '', 845),
       ( 'Celebrity Xpedition', '', 5530),
       ( 'Forse le Carib', '', 570),
       ( 'Porco Fugo', '', 240);

/*SELECT * FROM Transport;
SELECT * FROM Type_tickets;
SELECT * FROM Firms;*/

-- літаки(1-17), поїзда(51-60), кораблі(61-66), автобуси(18-50)
-- літаки(1-7), поїзда(8-11), кораблі(12-15), автобуси(16-22)
-- BEGIN;

INSERT INTO Transport_has_Type_tickets(transport_idtransport,
type_ticketa_idtype_tickets, count_seats_type)
VALUES (1, 7, 50),
       (1, 8, 75),
       (1, 9, 125),
       (2, 7, 30),
       (2, 8, 100),
       (2, 9, 40),
       (3, 7, 25),
       (3, 8, 75),
       (3, 9, 25),
       (4, 7, 45),
       (4, 8, 150),
       (4, 9, 50),
       (5, 7, 40),
       (5, 8, 200),
       (5, 9, 50),
       (6, 7, 30),
       (6, 8, 150),
       (6, 9, 50),
       (7, 7, 15),
       (7, 8, 80),
       (7, 9, 20),
       (8, 7, 14),
       (8, 8, 100),
       (8, 9, 20),
       (9, 7, 10),
       (9, 8, 50),
       (9, 9, 40),
       (10, 7, 20),
       (10, 8, 60),
       (10, 9, 30),
       (11, 7, 25),
       (11, 8, 200),
       (11, 9, 40),
       (12, 7, 80),
       (12, 8, 150),
       (12, 9, 100),
       (13, 7, 10),
       (13, 8, 75),
       (13, 9, 25),
       (14, 7, 35),
       (14, 8, 200),
       (14, 9, 50),
       (15, 7, 30),
```

```
(15, 8, 200),
(15, 9, 50),
(16, 7, 30),
(16, 8, 200),
(16, 9, 40),
(17, 7, 10),
(17, 8, 150),
(17, 9, 100),
(18, 10, 20),
(18, 11, 5),
(19, 10, 25),
(19, 11, 5),
(20, 10, 30),
(20, 11, 10),
(21, 10, 25),
(21, 11, 5),
(22, 10, 20),
(22, 11, 5),
(23, 10, 25),
(23, 11, 3),
(24, 10, 20),
(24, 11, 5),
(25, 10, 30),
(25, 11, 2),
(26, 10, 20),
(26, 11, 5),
(27, 10, 25),
(27, 11, 3),
(28, 10, 25),
(28, 11, 3),
(29, 10, 25),
(29, 11, 3),
(30, 10, 20),
(30, 11, 5),
(31, 10, 20),
(31, 11, 8),
(32, 10, 25),
(32, 11, 5),
(33, 10, 20),
(33, 11, 6),
(34, 10, 20),
(34, 11, 4),
(35, 10, 20),
(35, 11, 5),
(36, 10, 20),
(36, 11, 9),
(37, 10, 30),
(37, 11, 4),
(38, 10, 20),
(38, 11, 5),
(39, 10, 30),
(39, 11, 3),
(40, 10, 25),
(40, 11, 5),
(41, 10, 20),
(41, 11, 2),
(42, 10, 20),
(42, 11, 8),
(43, 10, 25),
(43, 11, 5),
(44, 10, 30),
(44, 11, 6),
(45, 10, 25),
```

```
(45, 11, 3),
(46, 10, 30),
(46, 11, 2),
(47, 10, 30),
(47, 11, 0),
(48, 10, 30),
(48, 11, 2),
(49, 10, 25),
(49, 11, 5),
(50, 10, 25),
(50, 11, 5),
(51, 1, 200),
(51, 2, 36),
(51, 3, 50),
(51, 4, 300),
(51, 5, 50),
(51, 6, 50),
(52, 1, 241),
(52, 2, 50),
(52, 3, 50),
(52, 4, 350),
(52, 5, 75),
(52, 6, 75),
(53, 1, 270),
(53, 2, 25),
(53, 3, 25),
(53, 4, 250),
(53, 5, 50),
(53, 6, 50),
(54, 1, 398),
(54, 2, 50),
(54, 3, 50),
(54, 4, 200),
(54, 5, 50),
(54, 6, 50),
(55, 1, 9),
(55, 2, 5),
(55, 3, 5),
(55, 4, 200),
(55, 5, 50),
(55, 6, 40),
(56, 1, 102),
(56, 2, 75),
(56, 3, 65),
(56, 4, 200),
(56, 5, 80),
(56, 6, 80),
(57, 1, 34),
(57, 2, 95),
(57, 3, 55),
(57, 4, 300),
(57, 5, 75),
(57, 6, 75),
(58, 1, 55),
(58, 2, 50),
(58, 3, 50),
(58, 4, 200),
(58, 5, 50),
(58, 6, 50),
(59, 1, 143),
(59, 2, 50),
(59, 3, 50),
(59, 4, 300),
```

```
        (59, 5, 50),
        (59, 6, 50),
        (60, 1, 142),
        (60, 2, 50),
        (60, 3, 50),
        (60, 4, 200),
        (60, 5, 50),
        (60, 6, 50),
        (61,12, 1500),
        (61,13, 405),
        (61,14, 500),
        (62,12, 750),
        (62,13, 250),
        (62,14, 300),
        (63,12, 500),
        (63,13, 145),
        (63,14, 200),
        (64,12, 4530),
        (64,13, 300),
        (64,14, 700),
        (65,12, 370),
        (65,13, 100),
        (65,14, 100),
        (66,12, 140),
        (66,13, 50),
        (66,14, 50);

-- SELECT * FROM Transport_has_Type_tickets;
-- ROLLBACK;

CREATE OR REPLACE FUNCTION calc_end_time() RETURNS trigger AS
$setEndTimeFlight$
    BEGIN
        CASE
            WHEN NEW.idFirm <= 7
                THEN NEW.endtime = NEW.starttime +
to_char(to_timestamp((NEW.f_distance/800.0) * 60.0), 'MI:SS')::INTERVAL;
            WHEN NEW.idFirm > 7 AND NEW.idFirm <= 11
                THEN NEW.endtime = NEW.starttime +
to_char(to_timestamp((NEW.f_distance/50.0) * 60.0), 'MI:SS')::INTERVAL;
            WHEN NEW.idFirm > 11 AND NEW.idFirm <= 15
                THEN NEW.endtime = NEW.starttime +
to_char(to_timestamp((NEW.f_distance/22.0) * 60.0), 'MI:SS')::INTERVAL;
            ELSE NEW.endtime = NEW.starttime +
to_char(to_timestamp((NEW.f_distance/65.0) * 60.0), 'MI:SS')::INTERVAL;
        END CASE;
        RETURN NEW;
    END;
$setEndTimeFlight$ LANGUAGE plpgsql;

-- DROP TRIGGER setEndTimeFlight ON Flights CASCADE;

CREATE TRIGGER setEndTimeFlight BEFORE INSERT ON Flights
   FOR EACH ROW EXECUTE PROCEDURE calc_end_time();


CREATE OR REPLACE FUNCTION validFlight() RETURNS trigger AS $checkDistance$
    BEGIN
        CASE
            WHEN NEW.f_distance <= 0
                THEN RAISE 'Invalid distance!';
            WHEN NEW.f_from ~ '[0-9]' OR NEW.f_to ~ '[0-9]'
                THEN RAISE 'This string can`t include digits!';
```

```
            ELSE RETURN NEW;
        END CASE;
    END;
$checkDistance$ LANGUAGE plpgsql;

-- DROP TRIGGER setEndTimeFlight ON Flights CASCADE;

CREATE TRIGGER checkDistance BEFORE INSERT ON Flights
   FOR EACH ROW EXECUTE PROCEDURE validFlight();

/*BEGIN ;
INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport, starttime)
VALUES (8, 'Київ', 'Дніпро', 488, 51, '2021-10-24 21:22:46');
ROLLBACK ;*/


CREATE OR REPLACE FUNCTION setStatus() RETURNS trigger AS $checkStatus$
    BEGIN
        CASE
             WHEN CURRENT_TIMESTAMP < NEW.endtime AND OLD.status = 'Відбувся'
                THEN NEW.status = 'Очікується';
            WHEN NEW.endtime < CURRENT_TIMESTAMP AND OLD.status =
'Очікується'
                THEN NEW.status = 'Відбувся';
            ELSE RETURN NEW;
        END CASE;
        RETURN NEW;
    END;
$checkStatus$ LANGUAGE plpgsql;

-- DROP TRIGGER setEndTimeFlight ON Flights CASCADE;

CREATE TRIGGER checkStatus BEFORE UPDATE OR INSERT ON Flights
   FOR EACH ROW EXECUTE PROCEDURE setStatus();


INSERT INTO Flights(idFirm, f_from, f_to, f_distance, idTransport, starttime)
VALUES (8, 'Київ', 'Дніпро', 488, 51, '2021-10-24 21:22:46'),
       (10, 'Херсон', 'Миколаїв', 80, 57, '2021-05-08 14:01:34'),
       (11, 'Київ', 'Луганськ', 922, 6, '2021-08-09 20:18:56'),
       (14, 'Одеса', 'Афіни', 1786, 61, '2021-06-15 00:12:53'),
       (9, 'Рівне', 'Луганськ', 1252, 56, '2021-09-13 14:36:39'),
       (2, 'Київ', 'Тирана', 1300, 5, '2021-01-08 21:21:34'),
       (3, 'Рівне', 'Монако', 1631, 15, '2021-04-21 00:32:33'),
       (1, 'Житомир', 'Люксембург', 1608, 8, '2021-10-20 17:46:29'),
       (1, 'Чернівці', 'Бухарест', 413, 16, '2021-09-15 10:20:25'),
       (8, 'Полтава', 'Миколаїв', 463, 52, '2021-09-05 16:22:17'),
       (10, 'Львів', 'Харків', 1109, 53, '2021-09-05 01:47:01'),
       (11, 'Черкаси', 'Львів', 727, 59, '2021-12-04 13:10:36'),
       (8, 'Ужгород', 'Чернівці', 432, 60, '2021-03-02 15:16:33'),
       (7, 'Донецьк', 'Мінськ', 945, 1, '2021-11-04 00:17:22'),
       (11, 'Хмельницький', 'Луганськ', 1247, 58, '2021-09-15 14:12:39'),
       (9, 'Івано-Франківськ', 'Вінниця', 381, 52, '2021-09-12 05:25:12'),
       (6, 'Івано-Франківськ', 'Валлетта', 1674, 17, '2021-09-19 12:36:24'),
       (1, 'Запоріжжя', 'Івано-Франківськ', 1147, 3, '2021-08-08 13:54:20'),
       (5, 'Київ', 'Вільнюс', 584, 9, '2021-11-05 17:59:33'),
       (1, 'Львів', 'Одеса', 623, 10, '2021-09-23 10:00:51'),
       (1, 'Суми', 'Івано-Франківськ', 1067, 12, '2021-09-12 11:39:37'),
       (22, 'Дніпро', 'Харків', 218, 50, '2021-10-24 19:04:23'),
       (19, 'Тернопіль', 'Рівне', 153, 48, '2021-07-10 19:58:31'),
       (18, 'Одеса', 'Мадрид', 3932, 43, '2021-08-20 16:54:35'),
       (17, 'Миколаїв', 'Хмельницький', 556, 31, '2021-06-23 10:10:45'),
       (19, 'Чернігів', 'Кропивницький', 426, 22, '2021-07-08 10:02:28'),
```

```
        (20, 'Вінниця', 'Чернігів', 411, 40, '2021-01-15 00:13:26'),
        (6, 'Кропивницький', 'Амстердам', 1985, 16, '2021-03-12 08:58:18'),
        (1, 'Луцьк', 'Любляна', 950, 17, '2021-02-23 19:18:07'),
        (3, 'Черкаси', 'Брюссель', 1964, 6, '2021-07-30 09:44:29'),
        (11, 'Житомир', 'Івано-Франківськ', 466, 55, '2021-07-21 12:52:50'),
        (7, 'Донецьк', 'Андорра-ла-Велья', 2845, 2, '2021-07-03 11:36:39'),
        (2, 'Хмельницький', 'Париж', 1788, 5, '2021-01-02 08:07:23'),
        (1, 'Луганськ', 'Суми', 267, 11, '2021-02-27 13:29:13'),
        (6, 'Суми', 'Гельсінкі', 1214, 15, '2021-03-15 02:15:15'),
        (1, 'Донецьк', 'Ужгород', 1333, 1, '2021-04-17 00:03:44'),
        (2, 'Харків', 'Бухарест', 976, 8, '2021-03-03 03:42:24'),
        (9, 'Івано-Франківськ', 'Суми', 1007, 54, '2021-03-16 08:17:25'),
        (21, 'Херсон', 'Київ', 545, 49, '2021-01-27 19:51:46'),
        (1, 'Ужгород', 'Кишинів', 673, 14, '2021-03-16 12:28:59'),
        (8, 'Херсон', 'Київ', 612, 58, '2021-01-12 18:07:00'),
        (2, 'Київ', 'Лісабон', 3355, 14, '2021-09-26 09:34:54'),
        (17, 'Тернопіль', 'Харків', 897, 34, '2021-02-22 01:45:39'),
        (1, 'Полтава', 'Софія', 1101, 7, '2021-01-14 01:33:36'),
        (19, 'Миколаїв', 'Київ', 479, 18, '2021-02-04 15:28:19'),
        (16, 'Житомир', 'Івано-Франківськ', 419, 23, '2021-03-09 18:59:38'),
        (1, 'Миколаїв', 'Сараєво', 1124, 6, '2021-10-06 18:55:15'),
        (22, 'Дніпро', 'Луцьк', 881, 39, '2021-10-21 07:47:48'),
        (10, 'Кропивницький', 'Миколаїв', 182, 53, '2021-03-10 02:42:48'),
        (3, 'Хмельницький', 'Берн', 1444, 4, '2021-03-19 11:52:21'),
        (1, 'Київ', 'Осло', 1630, 3, '2021-01-02 18:36:57');

-- SELECT * FROM Flights;

INSERT INTO Flights_has_Cities(idFlight, idCity1, idCity2)
(SELECT flight_ID, city_ID, (SELECT city_ID FROM Cities WHERE f_to =
city_name) FROM Flights, Cities
WHERE f_from = city_name);

-- SELECT * FROM Flights_has_Cities;

CREATE FUNCTION generate_air_ticket_price() RETURNS trigger AS
$setPriceAirTicket$
    BEGIN
        NEW.at_price = 4.0 * (SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
                on Flights.flight_ID = FhC.idFlight WHERE primaryID =
NEW.Flights_has_Cities_primaryID) +
            ((SELECT f_distance FROM Flights INNER JOIN Flights_has_Cities
FhC
                on Flights.flight_ID = FhC.idFlight WHERE idFlight =
NEW.Flights_has_Cities_primaryID) *
            (SELECT percent_afterpay/100.0 FROM Type_tickets
                WHERE NEW.idType_tickets = Type_tickets.typeticket_ID));
        RETURN NEW;
    END;
$setPriceAirTicket$ LANGUAGE plpgsql;

CREATE TRIGGER setPriceAirTicket BEFORE INSERT ON Airplane_tickets
   FOR EACH ROW EXECUTE PROCEDURE generate_air_ticket_price();
-- SELECT * FROM Airplane_tickets;

-- Запит для отримання ключів потрібних авірейсів з проміжної таблиці
/*SELECT idFirm, flight_ID, primaryID FROM Flights INNER JOIN
Flights_has_Cities
ON Flights.flight_ID = Flights_has_Cities.idFlight WHERE idFirm BETWEEN 1 AND
7;*/

INSERT INTO Airplane_tickets(idType_tickets, Flights_has_Cities_primaryID)
```

```
VALUES (7, 10),
       (9, 50),
       (9, 13),
       (8, 25),
       (8, 38),
       (7, 2),
       (8, 30),
       (9, 5),
       (8, 11),
       (7, 33),
       (8, 27),
       (8, 17),
       (8, 20),
       (9, 46),
       (9, 41),
       (7, 18),
       (8, 35),
       (8, 48),
       (9, 45),
       (8, 19),
       (7, 15),
       (8, 24),
       (8, 37),
       (9, 7),
       (8, 6);

-- SELECT * FROM Airplane_tickets;

CREATE FUNCTION set_air_class() RETURNS trigger AS $setClassAirHasCus$
    BEGIN
        CASE
            WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 7
                THEN NEW.at_class = 'Бізнес';
            WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 8
                THEN NEW.at_class = 'Економ';
            WHEN (SELECT idType_tickets FROM Airplane_tickets WHERE
NEW.Airplane_tickets_a_ticket_ID = a_ticket_ID) = 9
                THEN NEW.at_class = 'Перший';
        END CASE;
        RETURN NEW;
    END;
$setClassAirHasCus$ LANGUAGE plpgsql;

CREATE TRIGGER setClassAirHasCus BEFORE INSERT ON
Airplane_tickets_has_Customers
   FOR EACH ROW EXECUTE PROCEDURE set_air_class();


INSERT INTO Airplane_tickets_has_Customers(Airplane_tickets_a_ticket_ID,
Customers_customer_ID, a_booking_time,  a_seat, comments)
VALUES ( 1, 1, '2020-10-09 20:21:36', 1, NULL),
       ( 2, 2, '2020-01-14 09:55:01', 2, NULL),
       ( 3, 3, '2020-06-23 14:27:41', 3, NULL),
       ( 4, 4, '2020-05-26 03:05:58', 4, NULL),
       ( 5, 5, '2020-04-08 16:28:10', 5, NULL),
       ( 6, 6, '2020-04-14 02:14:58', 6, NULL),
       ( 7, 7, '2020-08-27 05:01:43', 7, NULL),
       ( 8, 8, '2020-09-17 00:14:03', 8, NULL),
       ( 9, 9, '2020-04-08 06:09:20', 9, NULL),
       ( 10, 10, '2020-04-07 05:52:12', 10, NULL),
       ( 11, 11, '2020-06-02 16:57:06', 11, NULL),
```

```
        ( 12, 12, '2020-06-01 14:13:55', 12, NULL),
        ( 13, 13, '2020-12-20 21:12:27', 13, NULL),
        ( 14, 14, '2020-04-30 03:00:09', 14, NULL),
        ( 15, 15, '2020-01-10 06:32:32', 15, NULL),
        ( 16, 16, '2020-10-19 16:27:48', 16, NULL),
        ( 17, 17, '2020-12-23 04:59:29', 17, NULL),
        ( 18, 18, '2020-03-08 17:26:36', 18, NULL),
        ( 19, 19, '2020-06-22 02:36:05', 19, NULL),
        ( 20, 20, '2020-02-10 15:28:49', 20, NULL),
        ( 21, 21, '2020-06-22 08:47:22', 21, NULL),
        ( 22, 22, '2020-12-17 13:17:31', 22, NULL),
        ( 23, 23, '2020-04-16 16:54:07', 23, NULL),
        ( 24, 24, '2020-06-25 04:22:31', 24, NULL),
        ( 25, 25, '2020-10-28 08:31:29', 25, NULL);



-- SELECT * FROM Customers;
-- SELECT * FROM Firms;
-- SELECT * FROM Transport;
-- SELECT * FROM Flights;
-- SELECT * FROM Flights_has_Cities;
-- SELECT * FROM Type_tickets;
-- SELECT * FROM Type_transportations;
-- SELECT * FROM Airplane_tickets;
-- SELECT * FROM Airplane_tickets_has_Customers;
-- SELECT * FROM Cities;


/*SELECT c_name, f_from, f_to, f_distance, at_price
FROM Customers INNER JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID INNER JOIN
Airplane_tickets A
ON AthC.Airplane_tickets_a_ticket_ID = A.a_ticket_ID INNER JOIN
Flights_has_Cities FhC
ON A.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Flights F
ON FhC.idFlight = F.flight_ID;*/


CREATE FUNCTION generate_ship_ticket_price() RETURNS trigger AS
$setPriceShipTicket$
    BEGIN
        NEW.st_price = 5.0 * (SELECT f_distance FROM Flights INNER JOIN
Flights_has_Cities FhC
                on Flights.flight_ID = FhC.idFlight WHERE primaryID =
NEW.Flights_has_Cities_primaryID) +
            ((SELECT f_distance FROM Flights INNER JOIN Flights_has_Cities
FhC
                on Flights.flight_ID = FhC.idFlight WHERE idFlight =
NEW.Flights_has_Cities_primaryID) *
            (SELECT percent_afterpay/100.0 FROM Type_tickets
                WHERE NEW.idType_tickets = Type_tickets.typeticket_ID));
        RETURN NEW;
    END;
$setPriceShipTicket$ LANGUAGE plpgsql;

CREATE TRIGGER setPriceShipTicket BEFORE INSERT ON Ship_tickets
   FOR EACH ROW EXECUTE PROCEDURE generate_ship_ticket_price();

-- SELECT * FROm Ship_tickets;

INSERT INTO Ship_tickets(idType_tickets, Flights_has_Cities_primaryID)
VALUES ( 12, 32),
```

```
       ( 13, 32),
       ( 14, 32),
       ( 12, 32),
       ( 13, 32),
       ( 14, 32),
       ( 12, 32),
       ( 13, 32),
       ( 14, 32),
       ( 12, 32),
       ( 13, 32),
       ( 14, 32),
       ( 12, 32),
       ( 13, 32),
       ( 14, 32),
       ( 12, 32),
       ( 13, 32),
       ( 14, 32),
       ( 12, 32),
       ( 13, 32),
       ( 12, 32),
       ( 12, 32),
       ( 13, 32),
       ( 12, 32),
       ( 12, 32),
       ( 13, 32),
       ( 12, 32),
       ( 12, 32),
       ( 13, 32),
       ( 14, 32),
       ( 12, 32),
       ( 13, 32),
       ( 13, 32),
       ( 13, 32),
       ( 13, 32),
       ( 13, 32);

-- SELECT * FROM Ship_tickets;

CREATE FUNCTION set_ship_class() RETURNS trigger AS $setClassShipHasCus$
    BEGIN
        CASE
            WHEN (SELECT idType_tickets
                    FROM Ship_tickets
                    WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 12
                THEN NEW.s_class = 'Стандарт';
            WHEN (SELECT idType_tickets
                    FROM Ship_tickets
                    WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 13
                THEN NEW.s_class = 'Преміум';
            WHEN (SELECT idType_tickets
                    FROM Ship_tickets
                    WHERE NEW.Ship_tickets_s_ticket_ID =
Ship_tickets.s_ticket_ID) = 14
                THEN NEW.s_class = 'Економ';
        END CASE;
        RETURN NEW;
    END;
$setClassShipHasCus$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER setClassShipHasCus BEFORE INSERT ON Ship_tickets_has_Customers
   FOR EACH ROW EXECUTE PROCEDURE set_ship_class();


INSERT INTO Ship_tickets_has_Customers(ship_tickets_s_ticket_id,
customers_customer_id, s_booking_time, s_seat, comments)
VALUES ( 1, 26, '2020-03-24 05:43:44', 23, NULL),
       ( 2, 27, '2020-02-25 13:56:49', 97, NULL),
       ( 3, 28, '2020-09-11 10:54:59', 45, NULL),
       ( 4, 29, '2020-11-19 02:10:09', 12, NULL),
       ( 5, 30, '2020-02-14 01:36:50', 88, NULL);

-- SELECT * FROM Ship_tickets_has_Customers;

INSERT INTO Routes(Flights_has_Cities_primaryID, idCity)
VALUES ( 8, 60),
       ( 8, 63),
       ( 8, 50),
       ( 29, 43),
       ( 29, 63),
       ( 39, 63),
       ( 39, 64),
       ( 39, 57),
       ( 39, 61),
       ( 31, 38),
       ( 31, 65),
       ( 31, 48),
       ( 40, 58),
       ( 51, 51),
       ( 51, 64),
       ( 51, 52),
       ( 28, 64),
       ( 28, 52),
       ( 28, 51),
       ( 1, 51),
       ( 1, 66),
       ( 43, 51),
       ( 43, 54),
       ( 43, 52),
       ( 43, 64),
       ( 4, 61),
       ( 3, 66),
       ( 3, 47),
       ( 3, 58),
       ( 3, 44),
--       маршрути поїздів
       ( 22, 57),
       ( 22, 45),
       ( 34, 52),
       ( 34, 54),
       ( 12, 65),
       ( 42, 51),
       ( 42, 54),
       ( 42, 52),
       ( 42, 64),
       ( 36, 47),
       ( 36, 51),
       ( 36, 57),
       ( 36, 61),
       ( 36, 55),
       ( 16, 43),
       ( 14, 51),
```

```
        ( 14, 63),
        ( 14, 59),
        ( 44, 54),
        ( 26, 51),
        ( 26, 57),
        ( 26, 61),
        ( 23, 54),
        ( 21, 57),
        ( 21, 61),
        ( 21, 55),
        ( 49, 60),
        ( 49, 63),
        ( 49, 53),
        ( 47, 64),
        ( 47, 55),
        ( 9, 50),
        ( 9, 60);

-- SELECT * FROM Routes;
-- SELECT * FROM Flights_has_Cities;
-- SELECT * FROM Flights;
-- SELECT * FROM Firms;
-- SELECT * FROM Cities WHERE country = 'Україна';

CREATE FUNCTION generate_bus_ticket_price() RETURNS trigger AS
$setPriceBusTicket$
    BEGIN
        NEW.bt_price = NEW.bt_price + NEW.bt_price*(SELECT
percent_afterpay/100.0
            FROM Type_tickets WHERE typeticket_ID = NEW.idType_tickets);
        RETURN NEW;
    END;
$setPriceBusTicket$ LANGUAGE plpgsql;

CREATE TRIGGER setPriceBusTicket BEFORE INSERT ON Bus_tickets
   FOR EACH ROW EXECUTE PROCEDURE generate_bus_ticket_price();


INSERT INTO Bus_tickets(bt_price, idType_tickets, idRoute)
VALUES ( 318, 10, 1),
       ( 212, 10, 2),
       ( 448, 10, 3),
       ( 430, 11, 4),
       ( 554, 10, 5),
       ( 112, 11, 6),
       ( 577, 10, 7),
       ( 638, 11, 8),
       ( 953, 10, 9),
       ( 3754, 10, 10),
       ( 496, 10, 11),
       ( 894, 10, 12),
       ( 153, 10, 13),
       ( 140, 11, 14),
       ( 297, 10, 15),
       ( 423, 10, 16),
       ( 310, 10, 17),
       ( 184, 10, 18),
       ( 384, 10, 19),
       ( 268, 10, 20),
       ( 409, 10, 21),
       ( 545, 10, 22),
       ( 66, 10, 23),
       ( 243, 10, 24),
```

```
       ( 368, 11, 25);

-- SELECT * FROM Bus_tickets;

INSERT INTO Bus_tickets_has_Customers(bus_tickets_b_ticket_id,
customers_customer_id, b_booking_time, b_seat)
VALUES ( 1, 26, '2020-05-08 02:11:23', 1),
       ( 2, 27, '2020-10-06 21:24:28', 4),
       ( 3, 28, '2020-05-14 23:59:25', 7),
       ( 4, 29, '2020-10-27 05:03:21', 19),
       ( 5, 30, '2020-09-07 08:19:49', 20),
       ( 6, 31, '2020-05-11 17:41:41', 14),
       ( 7, 32, '2020-10-30 03:44:05', 13),
       ( 8, 33, '2020-10-20 15:12:18', 7),
       ( 9, 34, '2020-02-11 14:04:45', 2),
       ( 10, 35, '2020-05-12 04:49:42', 10),
       ( 11, 36, '2020-01-20 08:54:35', 2),
       ( 12, 37, '2020-11-11 08:28:05', 10),
       ( 13, 38, '2020-10-21 01:13:59', 11),
       ( 14, 39, '2020-12-15 00:16:29', 12),
       ( 15, 40, '2020-07-09 17:08:32', 6),
       ( 16, 41, '2020-01-05 12:14:06', 17),
       ( 17, 42, '2020-08-14 14:08:17', 15),
       ( 18, 43, '2020-01-17 15:10:58', 4),
       ( 19, 44, '2020-11-29 22:43:53', 20),
       ( 20, 45, '2020-09-05 11:02:16', 2),
       ( 21, 46, '2020-02-12 22:26:31', 9),
       ( 22, 47, '2020-04-10 06:40:08', 8),
       ( 23, 48, '2020-01-24 19:43:14', 1),
       ( 24, 49, '2020-09-15 01:05:39', 5),
       ( 25, 50, '2020-07-22 09:01:15', 3);

-- SELECT * FROM Bus_tickets_has_Customers;

CREATE FUNCTION generate_train_ticket_price() RETURNS trigger AS
$setPriceTrainTicket$
    BEGIN
        NEW.tt_price = NEW.tt_price/2.0 + (NEW.tt_price/2.0)*(SELECT
percent_afterpay/100.0
            FROM Type_tickets WHERE typeticket_ID = NEW.idType_tickets);
        RETURN NEW;
    END;
$setPriceTrainTicket$ LANGUAGE plpgsql;

CREATE TRIGGER setPriceTrainTicket BEFORE INSERT ON Train_tickets
   FOR EACH ROW EXECUTE PROCEDURE generate_train_ticket_price();

INSERT INTO Train_tickets( tt_price, idType_tickets, idRoute)
VALUES ( 394, 1, 31),
       ( 554, 1, 32),
       ( 221, 2, 33),
       ( 404, 3, 34),
       ( 400, 4, 35),
       ( 545, 6, 36),
       ( 66, 3, 37),
       ( 243, 5, 38),
       ( 368, 2, 39),
       ( 188, 4, 40),
       ( 327, 4, 41),
       ( 720, 4, 42),
       ( 797, 4, 43),
       ( 1148, 5, 44),
       ( 364, 1, 45),
```

```
       ( 570, 2, 46),
       ( 241, 3, 47),
       ( 906, 4, 48),
       ( 66, 2, 49),
       ( 539, 2, 50),
       ( 931, 6, 51),
       ( 1009, 6, 52),
       ( 185, 1, 53),
       ( 394, 4, 54),
       ( 472, 5, 55);

-- SELECT * FROM Train_tickets;

INSERT INTO Train_tickets_has_Customers(train_tickets_t_ticket_id,
customers_customer_id, t_booking_time, t_numwagon, t_seat)
VALUES ( 1, 2, '2020-03-15 09:59:44', 1, 21),
       ( 2, 4, '2020-09-01 05:27:15', 6, 13),
       ( 3, 6, '2020-11-11 16:03:38', 4, 17),
       ( 4, 8, '2020-10-07 06:42:01', 9, 9),
       ( 5, 10, '2020-06-04 02:02:28', 10, 60),
       ( 6, 12, '2020-05-26 09:08:33', 3, 51),
       ( 7, 14, '2020-11-14 10:25:57', 2, 44),
       ( 8, 16, '2020-08-09 21:04:42', 7, 28),
       ( 9, 18, '2020-01-04 23:02:48', 1, 22),
       ( 10, 20, '2020-01-01 15:59:45', 3, 29),
       ( 11, 22, '2020-12-31 06:48:37', 6, 27),
       ( 12, 24, '2020-04-08 05:18:02', 7, 21),
       ( 13, 26, '2020-12-05 12:04:21', 2, 32),
       ( 14, 28, '2020-07-27 13:46:32', 8, 6),
       ( 15, 30, '2020-11-15 01:57:14', 9, 38),
       ( 16, 32, '2020-09-12 07:16:43', 10, 39),
       ( 17, 34, '2020-06-22 10:14:16', 8, 26),
       ( 18, 36, '2020-05-17 00:05:40', 7, 18),
       ( 19, 38, '2020-11-02 03:13:54', 4, 56),
       ( 20, 40, '2020-03-03 04:48:08', 2, 49),
       ( 21, 42, '2020-07-03 21:13:40', 3, 70),
       ( 22, 44, '2020-12-10 07:47:03', 5, 23),
       ( 23, 46, '2020-05-06 04:59:54', 7, 42),
       ( 24, 48, '2020-12-04 22:03:56', 1, 45),
       ( 25, 50, '2020-05-01 14:58:58', 2, 21);

-- SELECT * FROM Flights;


--ЗАПИТИ

-- 1. Фірми, що проводять тільки закордонні рейси і кількість цих рейсів
SELECT name, COUNT(flight_ID) AS Foreign_flights
FROM Flights INNER JOIN Firms F
ON Flights.idFirm = F.firm_ID INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE country <> 'Україна' AND NOT EXISTS( SELECT COUNT(flight_ID)
FROM Flights INNER JOIN Firms Fi
ON Flights.idFirm = Fi.firm_ID INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE country = 'Україна' AND F.name = Fi.name
HAVING COUNT(flight_ID) <> 0)
GROUP BY 1;


-- 2. Всі клієнти і їх вартості їхніх білетів на літаки, автобуси, поїзда
```

```
SELECT customer_ID, c_name,  at_price, Bt.bt_price, Tt.tt_price, st_price
FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
Airplane_tickets A
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID LEFT JOIN
Bus_tickets_has_Customers BthC
ON Customers.customer_ID = BthC.Customers_customer_ID LEFT JOIN Bus_tickets
Bt
ON Bt.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID LEFT JOIN
Train_tickets_has_Customers TthC
ON Customers.customer_ID = TthC.Customers_customer_ID LEFT JOIN Train_tickets
Tt
ON Tt.t_ticket_ID = TthC.Train_tickets_t_ticket_ID LEFT JOIN
Ship_tickets_has_Customers SthC
ON Customers.customer_ID = SthC.Customers_customer_ID LEFT JOIN Ship_tickets
S
ON S.s_ticket_ID = SthC.Ship_tickets_s_ticket_ID LEFT JOIN Flights_has_Cities
FhC
ON A.Flights_has_Cities_primaryID = FhC.primaryID LEFT JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets B
ON R.route_ID = B.idRoute LEFT JOIN Train_tickets T
ON R.route_ID = T.idRoute;

-- 3. Всі можливі маршрути до Києва з різних міст України
SELECT flight_ID, route_ID, f_from, city_name,idCity
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID INNER JOIN Cities C
ON C.city_ID = R.idCity
WHERE city_name = 'Київ' AND country = 'Україна';


-- 4. Фірми, що займаються повітряним або наземним перевезеннями та кількість
їхніх рейсів
SELECT F.name AS firm_name, type_name AS transition, COUNT(flight_ID) AS
number_of_flights
FROM Flights INNER JOIN Firms F
ON F.firm_ID = Flights.idFirm INNER JOIN Type_transportations Tt
ON Tt.transportation_ID = F.idTransportation
WHERE type_name IN ('Наземне', 'Повітряне')
GROUP BY 1, 2
HAVING COUNT(flight_ID) BETWEEN 3 AND 13;


-- 5. Авіаквитки куплені за останні 6 місяців
SELECT a_booking_time, at_price, at_class, a_seat
FROM Airplane_tickets INNER JOIN Airplane_tickets_has_Customers AthC
ON Airplane_tickets.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID
WHERE CURRENT_TIMESTAMP - INTERVAL '6 MONTH' < a_booking_time;


-- 6. Запит, що визначає тривалість рейсів, які виходять за межі України
SELECT CONCAT(f_from, ' - ', f_to) AS Flight,
       CONCAT(ROUND((EXTRACT(EPOCH FROM (endtime - starttime)) /
3600.00)::NUMERIC, 1), ' год.') AS Hours,
    CASE
        WHEN EXTRACT(EPOCH FROM (endtime - starttime)) / 3600.00 <= 3.0
            THEN 'Швидкий'
        WHEN EXTRACT(EPOCH FROM (endtime - starttime)) / 3600.00 <= 9.0
            THEN 'Середній'
        ELSE 'Тривалий'
    END AS Тривалість_рейсу
FROM Flights INNER JOIN Flights_has_Cities FhC
```

```
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE country <> 'Україна'
ORDER BY EXTRACT(EPOCH FROM (endtime - starttime)) / 3600.00 DESC;

-- SELECT f_from, f_to, f_distance, starttime, endtime, F.name, T.name
-- FROM Flights INNER JOIN Firms F on F.firm_ID = Flights.idFirm INNER JOIN
Transport T on T.transport_ID = Flights.idTransport
-- WHERE f_to = 'Мадрид';
-- SELECT * FROM Flights;

-- 7. Транспорт та квитки на нього
SELECT CONCAT(name, ' ', number), STRING_AGG(CONCAT(type_afterpay, ': ',
count_seats_type, ' шт.'), ' ')
FROM Transport INNER JOIN Transport_has_Type_tickets ThTt
ON Transport.transport_ID = ThTt.Transport_idTransport INNER JOIN
Type_tickets Tt
ON Tt.typeticket_ID = ThTt.Type_ticketa_idType_tickets
GROUP BY 1;


-- 8. Топ 3 найзбитковіші атобусні рейси
SELECT flight_ID, CONCAT(f_from, ' - ', f_to) AS Flight,
starttime::TIMESTAMP(0), endtime::TIMESTAMP(0),
       CONCAT(T.name, '  ', T.number) AS Bus_and_number,
       (SELECT COUNT(Bus_tickets_b_ticket_ID)
        FROM Bus_tickets_has_Customers INNER JOIN Bus_tickets Bt
        ON Bus_tickets_has_Customers.Bus_tickets_b_ticket_ID = Bt.b_ticket_ID
        INNER JOIN Routes R ON Bt.idRoute = R.route_ID INNER JOIN
Flights_has_Cities FhC
        ON R.Flights_has_Cities_primaryID = FhC.primaryID
        WHERE flight_ID = idFlight) AS Sold_tickets
FROM Flights INNER JOIN Firms F
ON Flights.idFirm = F.firm_ID INNER JOIN Type_transportations Tt
ON F.idTransportation = Tt.transportation_ID INNER JOIN Transport T
ON Flights.idTransport = T.transport_ID
WHERE type_name = 'Наземне'
ORDER BY 6
LIMIT 3;


-- 9. Пасажири що потратили найбільше грошей на покупку квитків на будь-який
вид перевезень
SELECT customer_ID, c_name, LPAD(CONCAT(ROUND(COALESCE(at_price, 0) +
COALESCE(Bt.bt_price, 0) +
                            COALESCE(Tt.tt_price, 0) + COALESCE(st_price, 0),
2), ' грн.'), 18, ' ') AS total
FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
Airplane_tickets A
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID LEFT JOIN
Bus_tickets_has_Customers BthC
ON Customers.customer_ID = BthC.Customers_customer_ID LEFT JOIN Bus_tickets
Bt
ON Bt.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID LEFT JOIN
Train_tickets_has_Customers TthC
ON Customers.customer_ID = TthC.Customers_customer_ID LEFT JOIN Train_tickets
Tt
ON Tt.t_ticket_ID = TthC.Train_tickets_t_ticket_ID LEFT JOIN
Ship_tickets_has_Customers SthC
ON Customers.customer_ID = SthC.Customers_customer_ID LEFT JOIN Ship_tickets
S
```

```
ON S.s_ticket_ID = SthC.Ship_tickets_s_ticket_ID LEFT JOIN Flights_has_Cities
FhC
ON A.Flights_has_Cities_primaryID = FhC.primaryID LEFT JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets B
ON R.route_ID = B.idRoute LEFT JOIN Train_tickets T
ON R.route_ID = T.idRoute
ORDER BY 3 DESC
LIMIT 10;


-- 10. Кількість маршрутів на кожному рейсі
SELECT type_name AS Transportation, CONCAT(f_from, ' - ', f_to) AS Flight,
COUNT(route_ID) AS number_of_routes,
       STRING_AGG(CONCAT(f_from, ' - ', city_name), ', ')
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets Bt
ON R.route_ID = Bt.idRoute LEFT JOIN Train_tickets Tt
ON R.route_ID = Tt.idRoute INNER JOIN Firms F
ON F.firm_ID = Flights.idFirm INNER JOIN Type_transportations T
ON T.transportation_ID = F.idTransportation INNER JOIN Cities C
ON C.city_ID = R.idCity
GROUP BY 1, 2
HAVING  1 < COUNT(route_ID)
ORDER BY 3 DESC;




-- ФУНКЦІЇ І ПРОЦЕДУРИ



-- 1. Функція знходження індекса останнього входження символа в строку
CREATE  OR REPLACE  FUNCTION LAST_POST(TEXT, CHAR)
RETURNS INTEGER
AS $$
    SELECT LENGTH($1) - LENGTH(REGEXP_REPLACE($1, '.*' || $2, ''));
$$ LANGUAGE SQL IMMUTABLE;

SELECT LAST_POST('I love PostgreSQL', 'e');

-- 2. Функція обчислення різниці між датами в днях
CREATE OR REPLACE FUNCTION DATEDIFF(timestamp, timestamp)
RETURNS INTEGER
AS $$
    SELECT ABS((CAST($1 AS date) - CAST($2 AS date))::INTEGER) as
DateDifference
$$ LANGUAGE SQL IMMUTABLE;

-- SELECT DATEDIFF('2000-01-01 00:00:00','2000-01-15 00:00:00');

-- 3. Функція пошуку авіквитків за ПІБ пасажира
CREATE OR REPLACE FUNCTION search_customer_tickets(search_name TEXT)
RETURNS SETOF text AS $$
DECLARE
    result TEXT DEFAULT '';
    records RECORD;
    myCursor CURSOR(search_name TEXT)
    FOR SELECT c_name AS Name, CONCAT(f_from, ' - ', f_to) AS Flight,
```

```
                        CONCAT(starttime, ' - ', endtime) AS Time, at_class
AS Class, a_seat AS Seat,
                         at_price AS Price, a_booking_time::TIMESTAMP(0) AS
Booking
                FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
                ON Customers.customer_ID = AthC.Customers_customer_ID LEFT
JOIN Airplane_tickets A
                ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID INNER
JOIN Flights_has_Cities FhC
                ON A.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN
Flights F
                ON FhC.idFlight = F.flight_ID;

BEGIN
    OPEN myCursor(search_name);-- відкриваєм курсор
    LOOP
    FETCH myCursor INTO records;
    EXIT WHEN NOT FOUND;
    IF records.Name = search_name
        THEN result := 'Пасажир: ' || records.Name || '  Рейс: ' ||
records.Flight || '  Час: (' || records.Time || ')  Клас: ' || records.Class
                           || '  Місце: ' || records.Seat || '  Ціна: ' ||
records.Price || '  Час бронювання: ' || records.Booking;
        RETURN NEXT result;
    END IF;
    END LOOP;
    CLOSE myCursor;-- закриваєм курсор
END;
$$ LANGUAGE plpgsql;


-- SELECT search_customer_tickets('Сомко Яромира Остапівна');
-- SELECT * FROM Customers INNER JOIN Airplane_tickets_has_Customers AthC on
Customers.customer_ID = AthC.Customers_customer_ID;
-- 4.
CREATE OR REPLACE FUNCTION update_status ()
RETURNS TABLE(flight_ID INT, idFirm INT, idTransport INT, from_city TEXT,
to_city TEXT, distance FLOAT, timeStart TIMESTAMP(0), timeEnd TIMESTAMP(0),
flight_status FLIGHTSTATUS)
AS $$
BEGIN
    RETURN QUERY
    UPDATE Flights
    SET status = 'Відбувся'
    WHERE endtime < CURRENT_TIMESTAMP AND status = 'Очікується'
    RETURNING *;
END;
$$ LANGUAGE plpgsql;

-- DROP FUNCTION update_status();

-- SELECT * FROM Flights;
-- BEGIN ;
-- SELECT * from update_status();
-- ROLLBACK ;

-- 5. Функція видалення
CREATE OR REPLACE FUNCTION auto_del(tableName TEXT, columnName TEXT, relOper
TEXT, value TEXT)
RETURNS VOID AS
$$
BEGIN
```

```
    EXECUTE 'DELETE FROM ' || tableName || ' WHERE ' || columnName || ' ' ||
relOper || E' \'' || value || E'\'';
END;
$$ LANGUAGE plpgsql VOLATILE;

/*BEGIN ;
SELECT auto_del('Customers', 'c_name', 'Тарасович Ратимир Вадимович');
ROLLBACK ;
SELECT * FROM Customers;*/


-- ---------------------------------------getStartTime-------------------------
-----------------
CREATE OR REPLACE FUNCTION getStartTime(TEXT, TEXT)
RETURNS TEXT
AS $$
    SELECT string_agg(starttime::TEXT, ', ')
    FROM Flights
    WHERE f_from = $1 AND f_to = $2
    ORDER BY 1
    LIMIT 1;
$$ LANGUAGE SQL IMMUTABLE;

CREATE OR REPLACE FUNCTION getStartTime(TEXT, TEXT, DATE, DATE)
RETURNS TEXT
AS $$
    SELECT string_agg(starttime::TEXT, ', ')
    FROM Flights
    WHERE f_from = $1 AND f_to = $2 AND (starttime BETWEEN $3::TIMESTAMP AND
$4::TIMESTAMP)
    ORDER BY 1
    LIMIT 1;
$$ LANGUAGE SQL IMMUTABLE;

-- DROP FUNCTION getStartTime(TEXT, TEXT);
-- DROP FUNCTION getStartTime(TEXT, TEXT, DATE, DATE);

-- ---------------------------------------------------------------------------
-----

/*SELECT f_from, f_to, starttime AS from_table, getStartTime(f_from, f_to,
'2021-01-01', '2021-09-01') AS from_function
FROM Flights
ORDER BY 1;*/

-- ---------------------------------------timeTable----------------------------
--------------
CREATE OR REPLACE FUNCTION timeTable()
RETURNS TABLE(Cities TEXT, "Івано-Франківськ" TEXT, Вінниця TEXT, Дніпро TEXT
, Донецьк TEXT, Житомир TEXT, Запоріжжя TEXT,
                              Київ TEXT, Кропивницький TEXT, Луганськ TEXT,
Луцьк TEXT, Львів TEXT,
                              Миколаїв TEXT, Одеса TEXT, Полтава TEXT, Рівне
TEXT, Суми TEXT,
                              Тернопіль TEXT, Ужгород TEXT, Харків TEXT, Херсон
TEXT, Хмельницький TEXT,
                               Черкаси TEXT, Чернівці TEXT, Чернігів TEXT)
AS $$
BEGIN
    RETURN QUERY
    SELECT DISTINCT city_name, getStartTime(city_name,'Івано-Франківськ') AS
"Івано-Франківськ",
```

```
                getStartTime(city_name,'Вінниця') AS "Вінниця",
getStartTime(city_name,'Дніпро') AS "Дніпро",
                getStartTime(city_name,'Донецьк') AS "Донецьк",
                getStartTime(city_name,'Житомир') AS "Житомир",
getStartTime(city_name,'Запоріжжя') AS "Запоріжжя",
                getStartTime(city_name,'Київ') AS "Київ",
getStartTime(city_name,'Кропивницький') AS "Кропивницький",
                getStartTime(city_name,'Луганськ') AS "Луганськ",
getStartTime(city_name,'Луцьк') AS "Луцьк",
                getStartTime(city_name,'Львів') AS "Львів",
getStartTime(city_name,'Миколаїв') AS "Миколаїв",
                getStartTime(city_name,'Одеса') AS "Одеса",
getStartTime(city_name,'Полтава') AS "Полтава",
                getStartTime(city_name,'Рівне') AS "Рівне",
getStartTime(city_name,'Суми') AS "Суми",
                getStartTime(city_name,'Тернопіль') AS "Тернопіль",
getStartTime(city_name,'Ужгород') AS "Ужгород",
                getStartTime(city_name,'Харків') AS "Харків",
getStartTime(city_name,'Херсон') AS "Херсон",
                getStartTime(city_name,'Хмельницький') AS "Хмельницький",
getStartTime(city_name,'Черкаси') AS "Черкаси",
                getStartTime(city_name,'Чернівці') AS "Чернівці",
getStartTime(city_name,'Чернігів') AS "Чернігів"
    FROM Flights LEFT JOIN Flights_has_Cities FhC on Flights.flight_ID =
FhC.idFlight
    INNER JOIN Cities C on C.city_ID = FhC.idCity1 OR C.city_ID = FhC.idCity2
    WHERE country = 'Україна'
    ORDER BY 1;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION timeTable(DATE, DATE)
RETURNS TABLE(Cities TEXT, "Івано-Франківськ" TEXT, Вінниця TEXT, Дніпро TEXT
, Донецьк TEXT, Житомир TEXT, Запоріжжя TEXT,
                            Київ TEXT, Кропивницький TEXT, Луганськ TEXT,
Луцьк TEXT, Львів TEXT,
                            Миколаїв TEXT, Одеса TEXT, Полтава TEXT, Рівне
TEXT, Суми TEXT,
                            Тернопіль TEXT, Ужгород TEXT, Харків TEXT, Херсон
TEXT, Хмельницький TEXT,
                             Черкаси TEXT, Чернівці TEXT, Чернігів TEXT)
AS $$
BEGIN
    IF $1 > $2
        THEN RAISE 'ERROR: timeTable(N DATE,M DATE); N must be <= M';
    ELSE RETURN QUERY
        SELECT DISTINCT city_name, getStartTime(city_name,'Івано-Франківськ',
$1, $2) AS "Івано-Франківськ",
                    getStartTime(city_name,'Вінниця', $1, $2) AS "Вінниця",
getStartTime(city_name,'Дніпро', $1, $2) AS "Дніпро",
                    getStartTime(city_name,'Донецьк', $1, $2) AS "Донецьк",
                    getStartTime(city_name,'Житомир', $1, $2) AS "Житомир",
getStartTime(city_name,'Запоріжжя', $1, $2) AS "Запоріжжя",
                    getStartTime(city_name,'Київ', $1, $2) AS "Київ",
getStartTime(city_name,'Кропивницький', $1, $2) AS "Кропивницький",
                    getStartTime(city_name,'Луганськ', $1, $2) AS "Луганськ",
getStartTime(city_name,'Луцьк', $1, $2) AS "Луцьк",
                    getStartTime(city_name,'Львів', $1, $2) AS "Львів",
getStartTime(city_name,'Миколаїв', $1, $2) AS "Миколаїв",
                    getStartTime(city_name,'Одеса', $1, $2) AS "Одеса",
getStartTime(city_name,'Полтава', $1, $2) AS "Полтава",
                    getStartTime(city_name,'Рівне', $1, $2) AS "Рівне",
getStartTime(city_name,'Суми', $1, $2) AS "Суми",
```

```
                    getStartTime(city_name,'Тернопіль', $1, $2) AS
"Тернопіль", getStartTime(city_name,'Ужгород', $1, $2) AS "Ужгород",
                    getStartTime(city_name,'Харків', $1, $2) AS "Харків",
getStartTime(city_name,'Херсон', $1, $2) AS "Херсон",
                    getStartTime(city_name,'Хмельницький', $1, $2) AS
"Хмельницький", getStartTime(city_name,'Черкаси', $1, $2) AS "Черкаси",
                    getStartTime(city_name,'Чернівці', $1, $2) AS "Чернівці",
getStartTime(city_name,'Чернігів', $1, $2) AS "Чернігів"
        FROM Flights LEFT JOIN Flights_has_Cities FhC on Flights.flight_ID =
FhC.idFlight
        INNER JOIN Cities C on C.city_ID = FhC.idCity1 OR C.city_ID =
FhC.idCity2
        WHERE country = 'Україна'
        ORDER BY 1;
    END IF;
END;
$$ LANGUAGE plpgsql;
-- ------------------------------------------------------------------------
-----
-- DROP FUNCTION timeTable();
-- DROP FUNCTION timeTable(DATE, DATE);
-- SELECT * FROM timeTable();
-- SELECT * FROM timeTable('2021-01-01', '2021-03-01');



-- ПРЕДСТАВЛЕННЯ

-- 1. Фірми, що проводять тільки закордонні рейси і кількість цих рейсів
CREATE VIEW foreign_firms AS SELECT firm_ID, name, COUNT(flight_ID) AS
amount_of_flights
FROM Flights INNER JOIN Firms F
ON Flights.idFirm = F.firm_ID INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE country <> 'Україна' AND NOT EXISTS( SELECT COUNT(flight_ID)
FROM Flights INNER JOIN Firms Fi
ON Flights.idFirm = Fi.firm_ID INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Cities C
ON C.city_ID = FhC.idCity2
WHERE country = 'Україна' AND F.name = Fi.name
HAVING COUNT(flight_ID) <> 0)
GROUP BY 1;


-- 2. Транспорт та квитки на нього
CREATE VIEW transport_tickets AS SELECT CONCAT(name, ' ', number),
STRING_AGG(CONCAT(type_afterpay, ': ', count_seats_type, ' шт.'), ' ')
FROM Transport INNER JOIN Transport_has_Type_tickets ThTt
ON Transport.transport_ID = ThTt.Transport_idTransport INNER JOIN
Type_tickets Tt
ON Tt.typeticket_ID = ThTt.Type_ticketa_idType_tickets
GROUP BY 1;


-- 3. Сума витрачиних коштів кожним пасажиром
CREATE VIEW customer_money AS SELECT customer_ID, c_name,
LPAD(CONCAT(ROUND(COALESCE(at_price, 0) +
                            COALESCE(Bt.bt_price, 0) + COALESCE(Tt.tt_price,
0) +
                            COALESCE(st_price, 0), 2), ' грн.'), 18, ' ') AS
total
```

```
FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
Airplane_tickets A
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID LEFT JOIN
Bus_tickets_has_Customers BthC
ON Customers.customer_ID = BthC.Customers_customer_ID LEFT JOIN Bus_tickets
Bt
ON Bt.b_ticket_ID = BthC.Bus_tickets_b_ticket_ID LEFT JOIN
Train_tickets_has_Customers TthC
ON Customers.customer_ID = TthC.Customers_customer_ID LEFT JOIN Train_tickets
Tt
ON Tt.t_ticket_ID = TthC.Train_tickets_t_ticket_ID LEFT JOIN
Ship_tickets_has_Customers SthC
ON Customers.customer_ID = SthC.Customers_customer_ID LEFT JOIN Ship_tickets
S
ON S.s_ticket_ID = SthC.Ship_tickets_s_ticket_ID LEFT JOIN Flights_has_Cities
FhC
ON A.Flights_has_Cities_primaryID = FhC.primaryID LEFT JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets B
ON R.route_ID = B.idRoute LEFT JOIN Train_tickets T
ON R.route_ID = T.idRoute
ORDER BY 3 DESC;


-- 4. Кількість маршрутів на кожному рейсі
CREATE VIEW routes_on_flight AS SELECT CONCAT(f_from, ' - ', f_to) AS Flight,
COUNT(route_ID) AS number_of_routes,
       STRING_AGG(CONCAT(f_from, ' - ', city_name), ', ')
FROM Flights INNER JOIN Flights_has_Cities FhC
ON Flights.flight_ID = FhC.idFlight INNER JOIN Routes R
ON FhC.primaryID = R.Flights_has_Cities_primaryID LEFT JOIN Bus_tickets Bt
ON R.route_ID = Bt.idRoute LEFT JOIN Train_tickets Tt
ON R.route_ID = Tt.idRoute INNER JOIN Firms F
ON F.firm_ID = Flights.idFirm INNER JOIN Type_transportations T
ON T.transportation_ID = F.idTransportation INNER JOIN Cities C
ON C.city_ID = R.idCity
GROUP BY 1
ORDER BY 2 DESC;

-- 5. Представлення, яке об`єднує найважливішу інформацію про авіаквитки
CREATE VIEW tickets_between_date AS SELECT c_name AS Name, CONCAT(f_from, ' -
', f_to) AS Flight,
        CONCAT(starttime, ' - ', endtime) AS Time, at_class AS Class, a_seat
AS Seat,
         at_price AS Price, a_booking_time::TIMESTAMP(0) AS Booking
FROM Customers LEFT JOIN Airplane_tickets_has_Customers AthC
ON Customers.customer_ID = AthC.Customers_customer_ID LEFT JOIN
Airplane_tickets A
ON A.a_ticket_ID = AthC.Airplane_tickets_a_ticket_ID INNER JOIN
Flights_has_Cities FhC
ON A.Flights_has_Cities_primaryID = FhC.primaryID INNER JOIN Flights F
ON FhC.idFlight = F.flight_ID INNER JOIN Firms F2
ON F.idFirm = F2.firm_ID INNER JOIN Type_transportations Tt
ON F2.idTransportation = Tt.transportation_ID
WHERE type_name = 'Повітряне';
-- КОРИСТУВАЧІ
CREATE USER Developer WITH PASSWORD 'Developer123';
GRANT ALL PRIVILEGES ON DATABASE Tickets_booking TO Developer;
CREATE USER Administrator WITH PASSWORD 'Administrator123';
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO Administrator;
CREATE USER Customer WITH PASSWORD 'Customer123';
GRANT SELECT ON tickets_booking.public.* TO Customer;
```