Technical University of Denmark

Mandatory Assignment

Airport Management Database

02170 Database Systems

Group 40

Josefine Rosalie Balch Petersen - s196671
Adrian Zvizdenco - s204683
Jeppe Møller Mikkelsen - s204708
Arthur Conrado Veiga Bosquetti - s204718
Marek Igor Fijolek - s214036

March 27, 2023

Contents

1 Statement of Requirements

We are modelling an airport with focus on the passengers and their movements within the airport. The database will be used to simulate the flow of passengers, depending on when flights arrive and depart, and what activities the passengers perform in the airport. The modelling of the database has been simplified for usage. Therefore, there is no concept of delayed flights, passengers not showing up, and lost luggage/passengers/flights.

The airport is organised into **terminals**, and each one of them consist of **gates** and **places**. The gates do not exist independently without a **terminal**, and are identified by an ID.

Places are predefined locations at the airport, such as food courts, tax-free shops, security checks, airline check-in counters etc.

The **flights** **stop** at **gates** and have **arrival/departure times** and **airport codes**. **Gates** are allocated for a certain **time period** to expected flights.

A **passenger** **holds** a **ticket** of certain **seating/boarding class**, which is their connection to a specific flight on **boarding**. The **passenger** may **own** one or more pieces of **luggage**.

The **luggage** is **owned** by the **passenger** and the **ticket** **tracks** the **luggage**.

A **flight** has a certain **capacity** of passengers and a certain **amount** of luggage. These capacities cannot be exceeded. We assume that the company never assigns too many passengers to a flight but they do not handle the luggage capacity. Therefore, in the case that too much luggage has been assigned a flight, the excess luggage will be reassigned to the next flight going to the same location.

The **ticket** is dependent on a **passenger** and permits a passenger to **board** a flight. The ticket is only valid for the certain **date**.

A **flight** **stops** at a **gate**, where it can either be arriving to or departing from. The flights also carry knowledge about their departure and arrival **airports**.

We allow the target airport access only to its inbound and outbound flights.

2 Conceptual Design

In order to present the Conceptual Design of our database, an **E-R Diagram** is displayed below on Figure 1; showing the entity sets as blue boxes with corresponding attributes and relationship sets as diamonds that can have attributes as well.

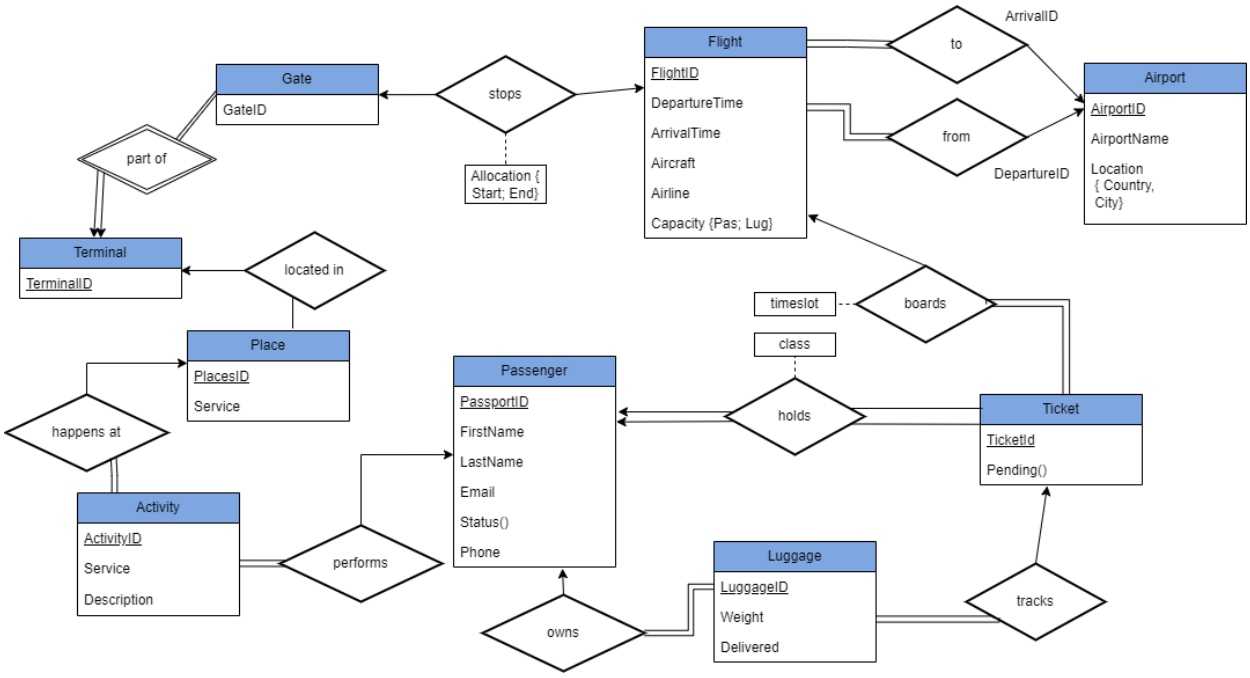

Figure 1: Entity-Relationship Diagram

In the Statement of Requirements we have colored the entities, relations and attributes in **red**, **blue** and **violet**, respectively.

The entity set **Airport** represents a set of airports with corresponding **code, name, location**, the former being a composite value having **country, city** as children. Not all airports need to have operating flights, therefore we chose partial participation in the relations with the entity **Flight**. On the other side, all flights need to have a certain source **airport** and destination **airport**, thus we use total participation. The relations **fly to** and **fly from** are `many-to-one`, since a flight can have only one source airport and only one destination airport, while several flights can be linked to a certain **airport**.

The entity set **Flight** represents a list of operated flights throughout the world. In order to limit the access only to flights that are related to our target airport, we will use the concept of table view. The flight has attributes **date, departure/arrival time, airline, aircraft, source/destination airport, capacity**, where capacity is a composite value of passenger number and luggage weight. In order to **board** a flight, a **ticket** should be valid on the specific **date** of the flight. All tickets should link to a flight, thus there is total participation for **Ticket** and partial participation for **Flight**, as flights may have no passengers at all. The Ticket-Flight relation is `many-to-one`, since a ticket can only link to one flight, and flights may have multiple tickets allocated to it. Also, approaching the **arrival/departure time** a **gate** is allocated such that the **flight** can stop there. The relation is `one-to-one`, since one flight needs only one gate and a gate can service only one flight, with partial participation as not all flights and not all gate may require an allocation at given time.

The weak entity set **Gate** is identified by a discriminator ID and the **terminal** they are located in, since different terminals may have the same gate identifier. Therefore, all gates (full participation) require to be **part of** a certain terminal and all terminals must have at least one gate in `many-to-one` relation. Several **places** with a **service type** may be **located in** a terminal in a `many-to-one` relation, since a certain location can be located in at most one terminal. The entity set **Activity** is identified by an ID and has a certain **description** and **type**, that has to be performed by one **passenger** in a specific **place**. Thus, in the two relations **happens at** and **performs** we use full participation for the **activity** and partial participation for the two entity sets. The relations are also `many-to-one` on the **activity** side, since many activities and many passengers can perform traceable actions.

The **Passenger** denotes people using the airport facilities having **first name, last name, email, phone** and a derived attribute **status** that is used to track whether the person and all its belongings (**luggage**) are arriving, departing or in transit. All **passengers** must **hold** one or multiple **tickets** and a ticket must be registered on a person, hence there is full participation from both entities in the `one-to-many` relation. Some passengers may **own** one or more pieces of **luggage**, in a `one-to-many` relation with full participation of **luggage**.

Finally, the **Ticket** can also **track** the **luggage** linked to it, the former having **ID, weight, status** denoting if it was successfully delivered or still waiting for boarding. All **luggage** must be linked to a certain ticket (full participation) in `many-to-one` relation. The **ticket** includes a derived attribute **pending()** that is active when there is tracked luggage that is awaiting transport to destination airport, the **ticket** being the one allowing the **boarding** of a plane.

3 Logical Design

Upon conversion of the conceptual design into a logical design the database schema diagram seen on figure 2 was obtained. The primary keys of each relation schema are underlined and foreign keys are depicted by arrows between two schemas.

Since **gate** is a weak entity set, in the logical design we have included the primary key of **terminal** in the relation attributes to form the primary key of **gate**. There's also a foreign key **FlightID** referencing the ID of the flight that is potentially allocated to the specific gate. The **Flight** schema has 2 **NOT NULL**-foreign keys **SourceCode, DestinationCode**, which references the primary key **AirportCode** obtained from the `many-to-one` relation conversion. The primary key consists of two attributes: **FlightID, Date** since we want to distinguish between flights on different days. **FlightID** is also referenced as a foreign key in the **Ticket** schema, since **Ticket** is on the many-side of the `many-to-one` relation. The primary key **TicketID** is referenced by **Luggage**, which is on the many side. Both **Ticket** and **Luggage** have a foreign key attribute referencing the primary key of **Passenger**. Finally, the **Activity** schema contains two foreign keys: **Place, Person** referencing the **PlaceID** as the location of the event and primary key **PassportID** as the subject of that event.

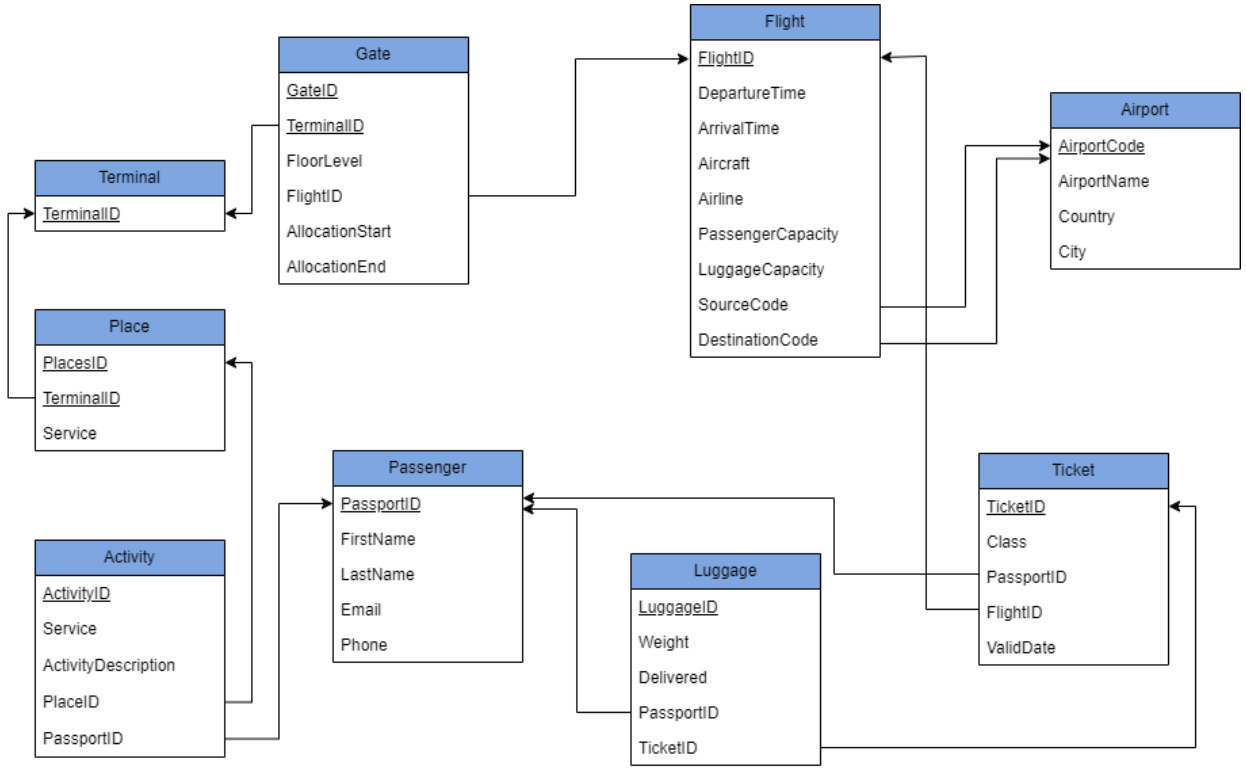

Figure 2: Relation Schema

4 Implementation

The implementation proved to an easy task, after having defined the relations schema in figure 2. However, some of the attribute names proved to be keywords in the SQL language, therefore, new names had to be found. The implementation resulted in a total of nine tables being created in the database AirportManagement.

In our implementation of CPH Airport Management, the table *Flight* is supposed to be seen as database storing all the flights around the world, and the ones CPH Airport can use/see is the flights related to CPH. Therefore, a **View** of the *Flight* where the flights either depart or arrive at CPH will be created and acted upon.

Constraints were used to define and control the context of the attributes, using the combination of the operators **Like**, **Regexp** and **Check** with basic string operators such as %-wildcard and Reg-Ex. Part of the core of our table creations is shown beneath, and rest of the table implementation can be found in the appendix 9.1.

```
# Flight schema
create table Flight
    (FlightID char(7) not null,
    constraint id_format check (FlightID regexp '^[A-Z]\{3,3\}[0-9]\{4,4\}$'),
    DepartureTime datetime,
    ArrivalTime datetime,
    Aircraft varchar(20),
    Airline varchar(20),
    PassengerCapacity decimal(3,0),
    LuggageCapacity decimal(4,0),
    SourceCode char(3),
    DestinationCode char(3),
    primary key(FlightID),
    foreign key(SourceCode) references Airport(AirportCode) on delete set null,
    foreign key(DestinationCode) references Airport(AirportCode) on delete set null
    );
```

```
# Passenger schema
create table Passenger
    (PassportID char(9) not null,
    FirstName varchar(20) not null,
    LastName varchar(30) null,
    Email varchar(40) not null,
    constraint email_format check (Email like '%@%'),
    Phone char(8),
    constraint code_format check (Phone regexp '^[0-9]*$'),
    primary key(PassportID)
    );
```

5 Database Instance

After populating the tables with corresponding data, the following tables have been obtained:

Passenger

PassportID	FirstName	LastName	Email	Phone
000000001	Arthur	Fosquetti	arthurfos@gmail.com	27305028
000000002	Jeppe	Snikkelsen	jepsnik@hotmail.com	17349228
000000003	Marek	Nijolek	mareknijolek@yahoo.com	29907628
000000004	Adrian	Snoozedenco	asnoozedenco@gmail.com	47569022
000000005	Rosalie	Batch	rosaliebatch@hotmail.com	23235514
000000006	Adrian	Palch	arthurbosq@gmail.com	27459121
000000007	Bianca	Onea	biaonea@hotmail.com	44789012
000000008	Sidse	Snomsen	sidses@yahoo.com	57801273
000000009	Yasmin	Fosquetti	slayfosquetti@gmail.com	33890923
000000010	Carina	Nijolek	carinanijo@hotmail.com	29461073
000000011	Paul	McCartney	pmccartney@hotmail.com	38399922
000000012	Snoop	Dogg	snoopydoggy@private.com	77832430
000000013	Jonas	Knocksen	pmccartney@hotmail.com	83901322
000000014	Marilyn	Monroe	marimonroe1@gmail.com	33997283
000000015	Neil	Armstrong	nielastro1@yahoo.com	66249129
000000016	John	Doe	johndoe123@hotmail.com	79823475
000000017	Jane	Door	door@hotmail.com	79866475
000000018	John	Smith	smithisthebest@hotmail.c...	79866575
000000019	Jane	Yuyu	janethejohn@hotmail.com	79866475
000000020	Yuri	Da Silva	yudas@gmail.com	12312345
000000021	Twoja	Stara	tokopara@gmail.com	21376942
000000022	Syn	Kolezanki	twojejmamy@gmail.com	2856301
000000023	Tonald	Drump	whitehouse@gmail.com	66699966

Ticket

TicketID	Class	PassportID	FlightID	TimeSlot
223EU89441637	Member	000000001	LAT2359	2023-04-22
223EU89441638	Member	000000004	SAS9921	2023-04-22
223EU89441639	Member	000000017	PLO2305	2023-04-22
420OL1722640I	First class	000000012	SNP0913	2023-04-22
420OL1722640J	Economy	000000020	HPE5320	2023-04-22
420OL1722640K	Economy	000000003	MAX1234	2023-04-22
4815162342LOS	Economy	000000009	MAL6692	2023-04-22
4815162342LOT	Economy	000000010	BAL6821	2023-04-22
8274SW1277464	Economy	000000016	POT5311	2023-04-22
98022MH1370X1	First class	000000015	MAL6666	2023-04-22
98022MH1370X2	Gold	000000018	XAM2432	2023-04-22
AUT2345623584	Member	000000023	GOL5021	2023-04-22
AUT5541903212	Gold	000000002	LAT6893	2023-04-22
AUT5541903213	First class	000000008	RYN5032	2023-04-22
AUT5541903214	First class	000000011	GOL5021	2023-04-22
AUT5541903215	Economy	000000014	BAL5326	2023-04-22
AUT5781398346	Economy	000000022	GOL5021	2023-04-22
AUT6729483818	Economy	000000021	GOL5021	2023-04-22
BV1938466382D	Gold	000000019	AMS1240	2023-04-22

Luggage

LuggageID	Weight	Delivered	PassportID	Ticket
16252410	17.30	0	000000004	223EU89441638
16252411	19.10	0	000000004	223EU89441638
16252412	5.20	0	000000004	223EU89441638
16372827	13.70	0	000000009	4815162342LOS
16627123	4.20	0	000000012	420OL1722640I
16627124	4.20	0	000000012	420OL1722640I
16627125	4.20	1	000000012	420OL1722640I
16627126	4.20	0	000000012	420OL1722640I
19287364	12.30	0	000000001	223EU89441637
19287365	18.40	0	000000001	223EU89441637
19287366	7.20	0	000000001	223EU89441637
23235541	11.20	0	000000008	AUT5541903213
45378284	14.10	0	000000022	AUT2345623584
72626221	3.20	0	000000010	4815162342LOT
76336251	14.50	0	000000007	AUT5541903212
77362513	19.30	0	000000015	98022MH1370X1
77362514	10.20	0	000000015	98022MH1370X1
87693432	16.60	0	000000016	8274SW1277464
92837234	19.40	0	000000003	420OL1722640K
98987772	10.20	0	000000011	AUT5541903214
98987773	17.40	0	000000014	AUT5541903214

Flight

FlightID	DepartureTime	ArrivalTime	Aircraft	Airline	PassengerCapacity	LuggageCapaci...	SourceCode	DestinationCo...
AMS1240	2023-03-22 09:30:00	2023-03-22 11:00:00	Boeing 767	Amsterdair	100	1500	CPH	AMS
BAL5326	2023-04-22 12:20:00	2023-04-22 15:30:00	Boeing 777	Ballights	100	1500	HND	HKG
BAL6821	2023-04-22 10:10:00	2023-04-22 13:20:00	Boeing 767	Ballights	100	1500	HKG	HND
GOL5021	2023-04-22 12:15:00	2023-04-22 13:35:00	Boeing 767	Gol	100	1500	GRU	FLN
HPE5320	2023-03-22 16:45:00	2023-03-22 09:30:00	Boeing 737	Hippie Flights	100	1500	CPH	HND
LAT2359	2023-04-22 10:50:00	2023-04-23 19:30:00	Boeing 737	Latam	100	1500	FLN	CPH
LAT6893	2023-04-22 19:30:00	2023-04-23 10:34:00	Boeing 767	Latam	100	1500	AMS	CPH
LOT2137	2023-04-22 12:05:00	2023-04-22 13:00:00	Airbus A380	LOT	560	9000	CPH	HKG
LOT9202	2023-04-22 12:05:00	2023-04-22 13:00:00	Airbus A320	LOT	90	1300	CPH	FLN
MAL6666	2023-04-22 06:05:00	2023-04-22 08:45:00	Boeing 777	Malokair	100	1500	CPH	GVA
MAL6692	2023-04-22 18:30:00	2023-04-22 19:34:00	Boeing 767	Malokair	100	1500	GVA	CPH
MAX1234	2023-04-22 12:00:00	2023-04-22 20:35:00	Boeing 777	Maxiflight	100	1500	FRA	CPH
PLG1245	2023-04-22 19:40:00	2023-04-23 05:50:00	Boeing 777	Pluggin Air	100	1500	CPH	JNB
PLO0239	2023-04-22 09:00:00	2023-04-22 11:40:00	Boeing 737	Plane On Air	100	1500	LUX	CPH
PLO2305	2023-04-22 23:25:00	2023-04-22 02:00:00	Boeing 737	Plane On Air	100	1500	CPH	LUX
POT5311	2023-04-22 15:55:00	2023-04-22 18:00:00	Boeing 767	Portugal Air	100	1500	CPH	LUX
RYN5032	2023-04-22 11:45:00	2023-04-23 10:30:00	Boeing 777	Rynair	100	1500	SYD	CPH
SAS9921	2023-04-22 11:45:00	2023-04-22 12:30:00	Boeing 767	Scandinavian	100	1500	AAL	CPH
SNP0913	2023-03-22 14:10:00	2023-03-22 23:10:00	Boeing 767	Snoop Lines	100	1500	JFK	CPH
XAM2432	2023-04-22 14:00:00	2023-04-22 16:00:00	Boeing 777	Xam Flights	100	1500	LHR	CPH

Gate

GateID	FlightID	AllocationSt...	AllocationEnd	FloorLevel	TerminalID
A10	LAT2359	NULL	NULL	0	1
A11	NULL	NULL	NULL	0	1
A33	SAS9921	NULL	NULL	1	1
B01	RYN5032	NULL	NULL	0	2
B02	NULL	NULL	NULL	1	2
C33	BAL5326	NULL	NULL	0	5

Terminal

TerminalID
1
2
3
4
5

Place

PlaceID	TerminalID	Service
P001	1	Duty-Free
P002	1	Check-In
P003	1	Check-Out
P004	1	Food
P005	1	Clothes
P006	2	Duty-Free
P007	2	Clothes
P008	2	Check-In
P009	2	Check-Out
P010	2	Books
P011	3	Duty-Free
P012	3	Perfume
P013	2	Clothes
P014	3	Check-In
P015	3	Check-Out
P016	3	Gifts
P017	4	Duty-Free
P018	4	News
P019	4	Check-In
P020	4	Check-Out
P021	5	Check-In
P022	5	Check-Out
P023	5	Duty-Free
P024	5	Food

Airport

AirportCo...	AirportName	Country	City
AAL	Aalborg Airport	Denmark	Aalborg
AMS	Amsterdam Airport Schipol	Netherlands	Amsterdam
CPH	Copenhagen International Airport	Denmark	Copenhagen
FLN	Aeroporto Internacional de Florianopolis	Brazil	Florianopolis
FRA	Frankfurt Airport	Germany	Frankfurt
GRU	Aeroporto Internacional de SP	Brazil	Guarulhos
GVA	Geneva Airport	Switzerland	Geneva
HKG	Hong Kong International Airport	Hong Kong	Chek Lap Kok
HND	Haneda Airport	Japan	Tokyo
JFK	John F. Kennedy International Airport	United States of America	New York
JNB	O.R. Tambo International Airport	South Africa	Johannesburg
LDZ	Lodz Airport	Poland	Lodz
LHR	Heathrow Airport	England	London
LUX	Luxembourg Airport	Luxembourg	Luxembourg
SYD	Sydney Kingsford Smith Airport	Australia	Sydney

Activity

ActivityID	Service	ActivityDescription	PlaceID	PassportID
A001	Check-In	Checked in for flight	P002	000000001
A002	Food	Bought a burger	P004	000000002
A003	Clothes	Bought a shirt	P005	000000003
A004	Check-Out	Checked out of the airport	P003	000000004
A005	Duty-Free	Bought a bottle of whiskey	P001	000000005
A006	Check-In	Checked in for flight	P008	000000006
A007	Books	Bought a book	P010	000000007
A008	Clothes	Bought on a pair of jeans	P007	000000008
A009	Check-Out	Checked out of the airport	P009	000000009
A010	Gifts	Bought a souvenir	P016	000000010
A011	Duty-Free	Bought a watch	P011	000000011
A012	Clothes	Bought on a shirt	P013	000000012
A013	Check-In	Checked in for the flight	P008	000000013
A014	Perfume	Bought a bottle of perfume	P012	000000014
A015	Food	Ordered a pizza	P004	000000015

Figure 3: Database Instance

The insert statements used to populate the database can be found in the appendix 9.2.

6 SQL Data Queries

In order to hide the general flight information from the Staff in our target airport, we decided to create a view of the flights that are related to the target CPH (inbound/outbound flights).

```
create user 'cphstaff'@'localhost' identified by 'hygge4ever';
grant all on AirportManagement.* to 'cphstaff'@'localhost';
revoke all on AirportManagement.Flight from 'cphstaff'@'localhost';
```

Therefore, the user we created has access to other tables than **Flight**. The created view (virtual table) selects the flights that have one of the SourceCode or DestinationCode equal to CPH.

```
create view CPHFlight as
    select * from Flight
        where sourceCode = "CPH" or destinationCode = "CPH";
grant all on AirportManagement.CPHFlight to 'cphstaff'@'localhost';
```

The view can be displayed using `select * from CPHFLight` and outputs the following 4:

FlightID	DepartureTime	ArrivalTime	Aircraft	Airline	PassengerCapacity	LuggageCapacity	SourceCode	DestinationCode
AMS1240	2023-03-22 09:30:00	2023-03-22 11:00:00	Boeing 767	Amsterdair	100	1500	CPH	AMS
HPE5320	2023-03-22 16:45:00	2023-03-22 09:30:00	Boeing 737	Hippie Flights	100	1500	CPH	HND
LAT2359	2023-04-22 10:50:00	2023-04-23 19:30:00	Boeing 737	Latam	100	1500	FLN	CPH
LAT6893	2023-04-22 19:30:00	2023-04-23 10:34:00	Boeing 767	Latam	100	1500	AMS	CPH
LOT2137	2023-04-22 12:05:00	2023-04-22 13:00:00	Airbus A380	LOT	560	9000	CPH	HKG
LOT9202	2023-04-22 12:05:00	2023-04-22 13:00:00	Airbus A320	LOT	90	1300	CPH	FLN
MAL6666	2023-04-22 06:05:00	2023-04-22 08:45:00	Boeing 777	Malokair	100	1500	CPH	GVA
MAL6692	2023-04-22 18:30:00	2023-04-22 19:34:00	Boeing 767	Malokair	100	1500	GVA	CPH
MAX1234	2023-04-22 12:00:00	2023-04-22 20:35:00	Boeing 777	Maxiflight	100	1500	FRA	CPH
PLG1245	2023-04-22 19:40:00	2023-04-23 05:50:00	Boeing 777	Pluggin Air	100	1500	CPH	JNB
PLO0239	2023-04-22 09:00:00	2023-04-22 11:40:00	Boeing 737	Plane On Air	100	1500	LUX	CPH
PLO2305	2023-04-22 23:25:00	2023-04-22 02:00:00	Boeing 737	Plane On Air	100	1500	CPH	LUX
POT5311	2023-04-22 15:55:00	2023-04-22 18:00:00	Boeing 767	Portugal Air	100	1500	CPH	LUX
RYN5032	2023-04-22 11:45:00	2023-04-23 10:30:00	Boeing 777	Rynair	100	1500	SYD	CPH
SAS9921	2023-04-22 11:45:00	2023-04-22 12:30:00	Boeing 767	Scandinavian	100	1500	AAL	CPH
SNP0913	2023-03-22 14:10:00	2023-03-22 23:10:00	Boeing 767	Snoop Lines	100	1500	JFK	CPH
XAM2432	2023-04-22 14:00:00	2023-04-22 16:00:00	Boeing 777	Xam Flights	100	1500	LHR	CPH

Figure 4: CPH Flight View

One important information that may be used by airline companies and similar stakeholders, is how the CPH Airport visitors are distributed and grouped into flight classes. We then made the following query to select this information from our database:

```
select Class, Count(PassportID) as MemberNo
    from CPHFlight natural join Passenger natural join Ticket
    group by Class
    order by MemberNo desc;
```

Class	MemberNo
Economy	4
Member	3
First class	3
Gold	3

Figure 5: Passenger Class

As mentioned in the statement of requirements, a flight cannot exceed its luggage weight capacity. It then becomes relevant for flight managers to know the luggage weight being carried on their flights. The following data query selects this information from our database:

```
select FlightID, SourceCode, ArrivalTime, DestinationCode, Sum(Weight) as TotalWeight
```

```
2       from Ticket natural join CPHFlight natural join Luggage
3       where DestinationCode = 'CPH'
4       group by FlightID;
```

FlightID	SourceCode	ArrivalTime	DestinationCode	TotalWeight
LAT2359	FLN	2023-04-23 19:30:00	CPH	37.90
MAL6692	GVA	2023-04-22 19:34:00	CPH	13.70
MAX1234	FRA	2023-04-22 20:35:00	CPH	19.40
RYN5032	SYD	2023-04-23 10:30:00	CPH	11.20
SAS9921	AAL	2023-04-22 12:30:00	CPH	41.60
SNP0913	JFK	2023-03-22 23:10:00	CPH	16.80

Figure 6: Luggage Weight on Arriving Flights

As it often happens, some luggage can be delayed. Airline companies can benefit significantly by knowing what passengers have delayed luggage and what their luggage is, so that they can put it on the next available flight. The query below selects this information from our database:

```
1  select FirstName, LastName, Email, LuggageID
2      from Passenger natural join Luggage natural join Flight natural join Ticket
3      where Delivered = false and DepartureTime < current_time();
```

FirstName	LastName	PassportID	Email	LuggageID
Snoop	Dogg	000000012	snoopydoggy@private.com	16627123
Snoop	Dogg	000000012	snoopydoggy@private.com	16627124
Snoop	Dogg	000000012	snoopydoggy@private.com	16627126

Figure 7: Delayed Luggage Owners

7 SQL Programming

It is often useful to know if there is a piece of luggage linked to a certain ticket that has been delayed and boarded a second flight in the same direction. Therefore, we have defined a `function` that takes as input a TicketID and checks if there exists a luggage entry that fulfills the predicate of being delayed:

```
1  delimiter //
2  create function PendingLuggage(tkt char(13)) returns boolean
3  begin
4  return exists (select * from Luggage
5          natural join Ticket natural join Flight
6              where TicketID = tkt and Delivered=false
7              and DepartureTime < current_time());
8  end; //
9  delimiter ;
```

In order to visualise the output of this function, one can run the function independently on a ticket value with `select PendingLuggage(<TicketID>)` or the function can be used as a column in a table query 8:

```
1  select TicketID, Class, PassportID, FlightID,
2      PendingLuggage(TicketID) as Pending from Ticket;
```

TicketID	Class	PassportID	FlightID	Pending
223EU89441637	Member	000000001	LAT2359	0
223EU89441638	Member	000000004	SAS9921	0
223EU89441639	Member	000000017	PLO2305	0
420OL1722640I	First class	000000012	SNP0913	1
420OL1722640J	Economy	000000020	HPE5320	0
420OL1722640K	Economy	000000003	MAX1234	0
4815162342LOS	Economy	000000009	MAL6692	0
4815162342LOT	Economy	000000010	BAL6821	0
8274SW1277464	Economy	000000016	POT5311	0
98022MH1370X1	First class	000000015	MAL6666	0
98022MH1370X2	Gold	000000018	XAM2432	0
AUT2345623584	Member	000000023	GOL5021	0
AUT5541903212	Gold	000000002	LAT6893	0
AUT5541903213	First class	000000008	RYN5032	0
AUT5541903214	First class	000000011	GOL5021	0
AUT5541903215	Economy	000000014	BAL5326	0
AUT5781398346	Economy	000000022	GOL5021	0
AUT6729483818	Economy	000000021	GOL5021	0
BV1938466382D	Gold	000000019	AMS1240	0

Figure 8: Tickets that link to Pending Luggage

As the departure/arrival time of each flight approaches, the staff members should be able to allocate an available gate at a terminal. Therefore, we have declared a procedure that performs the allocation if there are gates available and signals an error state, if all gates in the terminal selected are currently in use:

```
1  delimiter //
2  create procedure AllocateGate(IN flight char(7), IN terminal char(1))
3  begin
4      if (select GateID from Gate where TerminalID = terminal and FlightID is null) is null
5      then signal SQLSTATE 'HY000'
6          set mysql_errno = 1525, message_text='All gates are currently in use';
7      else
8          update Gate set FlightID = flight where TerminalID = terminal and FlightID is null
9          limit 1;
10     end if;
11 end; //
12 delimiter ;
```

Below, we'll present the table changes after applying the procedure twice:

```
1      call AllocateGate('POT5311', '1');
2      call AllocateGate('LOT2137', '2');
3      select * from Gate;
```

GateID	FlightID	AllocationStart	AllocationEnd	FloorLevel	TerminalID
A10	LAT2359	NULL	NULL	0	1
A11	NULL	NULL	NULL	0	1
A33	SAS9921	NULL	NULL	1	1
B01	RYN5032	NULL	NULL	0	2
B02	NULL	NULL	NULL	1	2
C33	BAL5326	NULL	NULL	0	5

Figure 9: Gates before allocation

GateID	FlightID	AllocationStart	AllocationEnd	FloorLevel	TerminalID
A10	LAT2359	NULL	NULL	0	1
A11	POT5311	NULL	NULL	0	1
A33	SAS9921	NULL	NULL	1	1
B01	RYN5032	NULL	NULL	0	2
B02	NULL	NULL	NULL	1	2
C33	BAL5326	NULL	NULL	0	5

Figure 10: Gates after first allocation

GateID	FlightID	AllocationStart	AllocationEnd	FloorLevel	TerminalID
A10	LAT2359	NULL	NULL	0	1
A11	POT5311	NULL	NULL	0	1
A33	SAS9921	NULL	NULL	1	1
B01	RYN5032	NULL	NULL	0	2
B02	LOT2137	NULL	NULL	1	2
C33	BAL5326	NULL	NULL	0	5

Figure 11: Gates after 2 allocations

In case all gates in the specified terminal are currently in use the following error is signaled as output:

```
Error Code 1525:  All gates are currently in use!
```

Finally, when inserting new rows into the Flight table, the staff should be able to check the temporal sequence of the Arrival and Departure time and the fact that the source and destination airports are different. In order to check for those rules, we have declared a trigger that signals errors in case of any rule violation:

```
1      delimiter //
2      create trigger CheckTimeAirport
3      before insert on Flight for each row
4      begin
5          if new.ArrivalTime <= new.DepartureTime
6          then signal sqlstate 'HY000'
7              set mysql_errno = 1580,
8              message_text = 'Arrival time should be later than departure time';
9      end if;
10     if new.SourceCode = new.DestinationCode
11         then signal sqlstate 'HY000'
12             set mysql_errno = 1590,
13                 message_text = 'A flight cannot have the same
14                         source and destination airport';
15     end if;
16     end //
17     delimiter ;
```

8 SQL Table Modifications

An update to redirect flights with a certain destination has been redirected through CPH Airport. Here we have flights to Honk-Kong and Florianopolis with (`DestinationCode` 'FLN' and 'HKG') to CPH Airport redirected, unless they already departed from Copenhagen, which in that case the flight would not be considered:

```
1  update Flight
2  set DestinationCode = 'CPH'
3  where SourceCode != 'CPH' and DestinationCode in ('HKG', 'FLN');
```

We can see in Figure 12 that CPHFlight has two more entries than in Figure 4, since these flights now are related to CPH Airport.

FlightID	DepartureTime	ArrivalTime	Aircraft	Airline	PassengerCapacity	LuggageCapacity	SourceCode	DestinationCode
AMS1240	2023-03-22 09:30:00	2023-03-22 11:00:00	Boeing 767	Amsterdair	100	1500	CPH	AMS
BAL5326	2023-04-22 12:20:00	2023-04-22 15:30:00	Boeing 777	Ballights	100	1500	HND	CPH
GOL5021	2023-04-22 12:15:00	2023-04-22 13:35:00	Boeing 767	Gol	100	1500	GRU	CPH
HPE5320	2023-03-22 16:45:00	2023-03-22 09:30:00	Boeing 737	Hippie Flights	100	1500	CPH	HND
LAT2359	2023-04-22 10:50:00	2023-04-23 19:30:00	Boeing 737	Latam	100	1500	FLN	CPH
LAT6893	2023-04-22 19:30:00	2023-04-23 10:34:00	Boeing 767	Latam	100	1500	AMS	CPH
LOT2137	2023-04-22 12:05:00	2023-04-22 13:00:00	Airbus A380	LOT	560	9000	CPH	HKG
LOT9202	2023-04-22 12:05:00	2023-04-22 13:00:00	Airbus A320	LOT	90	1300	CPH	FLN
MAL6666	2023-04-22 06:05:00	2023-04-22 08:45:00	Boeing 777	Malokair	100	1500	CPH	GVA
MAL6692	2023-04-22 18:30:00	2023-04-22 19:34:00	Boeing 767	Malokair	100	1500	GVA	CPH
MAX1234	2023-04-22 12:00:00	2023-04-22 20:35:00	Boeing 777	Maxiflight	100	1500	FRA	CPH
PLG1245	2023-04-22 19:40:00	2023-04-23 05:50:00	Boeing 777	Pluggin Air	100	1500	CPH	JNB
PLO0239	2023-04-22 09:00:00	2023-04-22 11:40:00	Boeing 737	Plane On Air	100	1500	LUX	CPH
PLO2305	2023-04-22 23:25:00	2023-04-22 02:00:00	Boeing 737	Plane On Air	100	1500	CPH	LUX
POT5311	2023-04-22 15:55:00	2023-04-22 18:00:00	Boeing 767	Portugal Air	100	1500	CPH	LUX
RYN5032	2023-04-22 11:45:00	2023-04-23 10:30:00	Boeing 777	Rynair	100	1500	SYD	CPH
SAS9921	2023-04-22 11:45:00	2023-04-22 12:30:00	Boeing 767	Scandinavian	100	1500	AAL	CPH
SNP0913	2023-03-22 14:10:00	2023-03-22 23:10:00	Boeing 767	Snoop Lines	100	1500	JFK	CPH
XAM2432	2023-04-22 14:00:00	2023-04-22 16:00:00	Boeing 777	Xam Flights	100	1500	LHR	CPH

Figure 12: CPHFlight view after updating Flights table

An update statement for upgrading all economy-class passengers on flight GOL5021 without luggage, to 'First Class'. This is done with the following query:

```
1  update Ticket left join Luggage
2  on Luggage.PassportID = Ticket.PassportID
3  set Class = 'First class'
4  where Luggage.PassportID is null and
5      Ticket.FlightID = 'GOL5021' and Ticket.Class = 'Economy';
```

As we can see in Figure 13 when compared against 3, only one row was affected, which is correct. All other passengers of this flight either have a higher class or have a luggage (or both), as seen on 13.

TicketID	Class	PassportID	FlightID	TimeSlot
223EU89441637	Member	000000001	LAT2359	2023-04-22
223EU89441638	Member	000000004	SAS9921	2023-04-22
223EU89441639	Member	000000017	PLO2305	2023-04-22
420OL1722640I	First class	000000012	SNP0913	2023-04-22
420OL1722640J	Economy	000000020	HPE5320	2023-04-22
420OL1722640K	Economy	000000003	MAX1234	2023-04-22
4815162342LOS	Economy	000000009	MAL6692	2023-04-22
4815162342LOT	Economy	000000010	BAL6821	2023-04-22
8274SW1277464	Economy	000000016	POT5311	2023-04-22
98022MH1370X1	First class	000000015	MAL6666	2023-04-22
98022MH1370X2	Gold	000000018	XAM2432	2023-04-22
AUT2345623584	Member	000000023	GOL5021	2023-04-22
AUT5541903212	Gold	000000002	LAT6893	2023-04-22
AUT5541903213	First class	000000008	RYN5032	2023-04-22
AUT5541903214	First class	000000011	GOL5021	2023-04-22
AUT5541903215	Economy	000000014	BAL5326	2023-04-22
AUT5781398346	Economy	000000022	GOL5021	2023-04-22
AUT6729483818	First class	000000021	GOL5021	2023-04-22
BV1938466382D	Gold	000000019	AMS1240	2023-04-22

Figure 13: Ticket table after upgrading luggage-less economy passengers of flight GOL5021

The next query we made was a delete-statement, which removes all activities which happened in the terminal number 3. To achieve this goal, we needed to join the Activity table with Place table. The query can be seen below:

```
1  delete a from Activity a
2  join Place on a.PlaceID = Place.PlaceID
3  where Place.TerminalID = '3';
```

Another update-statement was made to upgrade all passengers tickets by one class (such that Economy tickets were upgrade to Member, Member to First class, and First class to Gold). The SQL query:

```
update Ticket set Class =
    case
    when class = 'Gold' then class
    when class = 'First class' then 'Gold'
    when class = 'Member' then 'First class'
    when class = 'Economy' then 'Member'
    end;
```

The updated Ticket-table can be seen in Figure 14. Naturally, there are no more Economy class tickets.

TicketID	Class	PassportID	FlightID	TimeSlot
223EU89441637	First class	000000001	LAT2359	2023-04-22
223EU89441638	First class	000000004	SAS9921	2023-04-22
223EU89441639	First class	000000017	PLO2305	2023-04-22
420OL1722640I	Gold	000000012	SNP0913	2023-04-22
420OL1722640J	Member	000000020	HPE5320	2023-04-22
420OL1722640K	Member	000000003	MAX1234	2023-04-22
4815162342LOS	Member	000000009	MAL6692	2023-04-22
4815162342LOT	Member	000000010	BAL6821	2023-04-22
8274SW1277464	Member	000000016	POT5311	2023-04-22
98022MH1370X1	Gold	000000015	MAL6666	2023-04-22
98022MH1370X2	Gold	000000018	XAM2432	2023-04-22
AUT5541903212	Gold	000000002	LAT6893	2023-04-22
AUT5541903213	Gold	000000008	RYN5032	2023-04-22
AUT5541903214	Gold	000000011	GOL5021	2023-04-22
AUT5541903215	Member	000000014	BAL5326	2023-04-22
BV1938466382D	Gold	000000019	AMS1240	2023-04-22

Figure 14: Ticket table after all tickets were upgraded by one class

Furthermore, we used the delete statement to simulate cancelling all flights from airline 'Malokair' for the given day using: `delete from flight where airline='malokair';`. Figure 15 shows the resulting Flights table after such deletion:

FlightID	DepartureTime	ArrivalTime	Aircraft	Airline	PassengerCapacity	LuggageCapacity	SourceCode	DestinationCode
AMS1240	2023-03-22 09:30:00	2023-03-22 11:00:00	Boeing 767	Amsterdair	100	1500	CPH	AMS
BAL5326	2023-04-22 12:20:00	2023-04-22 15:30:00	Boeing 777	Ballights	100	1500	HND	CPH
BAL6821	2023-04-22 10:10:00	2023-04-22 13:20:00	Boeing 767	Ballights	100	1500	HKG	HND
GOL5021	2023-04-22 12:15:00	2023-04-22 13:35:00	Boeing 767	Gol	100	1500	GRU	CPH
HPE5320	2023-03-22 16:45:00	2023-03-22 09:30:00	Boeing 737	Hippie Flights	100	1500	CPH	HND
LAT2359	2023-04-22 10:50:00	2023-04-23 19:30:00	Boeing 737	Latam	100	1500	FLN	CPH
LAT6893	2023-04-22 19:30:00	2023-04-23 10:34:00	Boeing 767	Latam	100	1500	AMS	CPH
LOT2137	2023-04-22 12:05:00	2023-04-22 13:00:00	Airbus A380	LOT	560	9000	CPH	HKG
LOT9202	2023-04-22 12:05:00	2023-04-22 13:00:00	Airbus A320	LOT	90	1300	CPH	FLN
MAX1234	2023-04-22 12:00:00	2023-04-22 20:35:00	Boeing 777	Maxiflight	100	1500	FRA	CPH
PLG1245	2023-04-22 19:40:00	2023-04-23 05:50:00	Boeing 777	Pluggin Air	100	1500	CPH	JNB
PLO0239	2023-04-22 09:00:00	2023-04-22 11:40:00	Boeing 737	Plane On Air	100	1500	LUX	CPH
PLO2305	2023-04-22 23:25:00	2023-04-22 02:00:00	Boeing 737	Plane On Air	100	1500	CPH	LUX
POT5311	2023-04-22 15:55:00	2023-04-22 18:00:00	Boeing 767	Portugal Air	100	1500	CPH	LUX
RYN5032	2023-04-22 11:45:00	2023-04-23 10:30:00	Boeing 777	Rynair	100	1500	SYD	CPH
SAS9921	2023-04-22 11:45:00	2023-04-22 12:30:00	Boeing 767	Scandinavian	100	1500	AAL	CPH
SNP0913	2023-03-22 14:10:00	2023-03-22 23:10:00	Boeing 767	Snoop Lines	100	1500	JFK	CPH
XAM2432	2023-04-22 14:00:00	2023-04-22 16:00:00	Boeing 777	Xam Flights	100	1500	LHR	CPH

Figure 15: Flights table after removing flights from 'Malokair'

9 Appendix

9.1 Implementation

```
# ___________________SCHEMA IMPLEMENTATION___________________
# Terminal schema
create table Terminal(
    TerminalID char(1) not null,
    constraint id_format check (TerminalID regexp '^[1-9]$'),
```

```
6       primary key(TerminalID)
7       );
8
9   # Airports schema
10  create table Airport
11      (AirportCode char(3),
12      constraint code_format check (AirportCode regexp '^[A-Z]\{3,3\}$'),
13      AirportName varchar(50),
14      Country varchar(40) not null,
15      City varchar(40),
16      primary key(AirportCode)
17      );
18
19  # Passenger schema
20  create table Passenger
21      (PassportID char(9) not null,
22      FirstName varchar(20) not null,
23      LastName varchar(30) null,
24      Email varchar(40) not null,
25      constraint email_format check (Email like '%@%'),
26      Phone char(8),
27      constraint code_format check (Phone regexp '^[0-9]*$'),
28      primary key(PassportID)
29      );
30
31  # Place schema
32  create table Place
33      (PlaceID varchar(4) not null,
34      TerminalID char(1) not null,
35      Service varchar(10),
36      foreign key(TerminalID) references Terminal(TerminalID) on delete cascade,
37      primary key (PlaceID)
38      );
39
40  # Activity schema
41  create table Activity
42      (ActivityID varchar(4),
43      Service varchar(10),
44      ActivityDescription varchar(40),
45      PlaceID varchar(4) not null,
46      PassportID char(9) not null,
47      foreign key(PlaceID) references Place(PlaceID) on delete cascade,
48      foreign key(PassportID) references Passenger(PassportID) on delete cascade,
49      primary key(ActivityID)
50      );
51
52  # Flight schema
53  create table Flight
54      (FlightID char(7) not null,
55      constraint id_format check (FlightID regexp '^[A-Z]\{3,3\}[0-9]\{4,4\}$'),
56      DepartureTime datetime,
57      ArrivalTime datetime,
58      Aircraft varchar(20),
59      Airline varchar(20),
60      PassengerCapacity decimal(3,0),
61      LuggageCapacity decimal(4,0),
62      SourceCode char(3),
63      DestinationCode char(3),
64      primary key(FlightID),
65      foreign key(SourceCode) references Airport(AirportCode) on delete set null,
66      foreign key(DestinationCode) references Airport(AirportCode) on delete set null
67      );
68
69
70  # Gate schema
```

```
71  create table Gate(
72      GateID varchar(3) not null,
73      FlightID char(7),
74      AllocationStart Time,
75      AllocationEnd Time,
76      FloorLevel decimal(1,0),
77      TerminalID char(1) not null,
78      constraint id_format check (GateID regexp '^[A-Z][0-9][0-9]$'),
79      foreign key(TerminalID) references Terminal(TerminalID) on delete cascade,
80      foreign key(FlightID) references Flight(FlightID) on delete set null,
81      primary key(GateID, TerminalID)
82      );
83
84  # Ticket schema
85  create table Ticket
86      (TicketID char(13) not null,
87      Class ENUM('First class','Gold', 'Member', 'Economy'),
88      PassportID char(9) not null,
89      FlightID char(10),
90      TimeSlot date,
91      foreign key(PassportID) references Passenger(PassportID) on delete cascade,
92      foreign key(FlightID) references Flight(FlightID) on delete set null,
93      primary key(TicketID)
94      );
95
96  # Luggage schema
97  create table Luggage
98      (LuggageID char(8) not null,
99      Weight decimal (4,2) not null,
100     Delivered bool,
101     PassportID char(9) not null,
102     Ticket char(13) not null,
103     primary key(LuggageID),
104     foreign key(Ticket) references Ticket(TicketID),
105     foreign key(PassportID) references Passenger(PassportID),
106     constraint code_format check (LuggageID regexp '^[0-9]*$')
107     );
```

9.2 Populating the database

```
1
2  # ____________________DATABASE POPULATION_____________________
3  insert into Terminal(TerminalID) values ('1'), ('2'), ('3'), ('4'), ('5');
4
5  # Airport inserts
6  insert into Airport values
7      ('CPH', 'Copenhagen International Airport', 'Denmark', 'Copenhagen'),
8      ('FLN', 'Aeroporto International de Florianopolis', 'Brazil', 'Florianopolis'),
9      ('AMS', 'Amsterdam Airport Schipol', 'Netherlands', 'Amsterdam'),
10     ('FRA', 'Frankfurt Airport', 'Germany', 'Frankfurt'),
11     ('GRU', 'Aeroporto Internacional de SP', 'Brazil', 'Guarulhos'),
12     ('SYD', 'Sydney Kingsford Smith Airport', 'Australia', 'Sydney'),
13     ('LDZ', 'Lodz Airport', 'Poland', 'Lodz'),
14     ('GVA', 'Geneva Airport', 'Switzerland', 'Geneva'),
15     ('HKG', 'Hong Kong International Airport', 'Hong Kong', 'Chek Lap Kok'),
16     ('AAL', 'Aalborg Airport', 'Denmark', 'Aalborg'),
17     ('JFK', 'John F. Kennedy International Airport', 'United States of America', 'New York'),
18     ('LUX', 'Luxembourg Airport', 'Luxembourg', 'Luxembourg'),
19     ('LHR', 'Heathrow Airport', 'England', 'London'),
20     ('HND', 'Haneda Airport', 'Japan', 'Tokyo'),
21     ('JNB', 'O.R. Tambo International Airport', 'South Africa', 'Johannesburg');
22
```

```
23 # Passenger inserts
24 insert Passenger values
25     # PassportID, FirstName, LastName, Email, Phone
26     ('000000001','Arthur','Fosquetti','arthurfos@gmail.com','27305028'),
27     ('000000002','Jeppe','Snikkelsen','jepsnik@hotmail.com','17349228'),
28     ('000000003','Marek','Nijolek','mareknijolek@yahoo.com','29907628'),
29     ('000000004','Adrian','Snoozedenco ','asnoozedenco@gmail.com','47569022'),
30     ('000000005','Rosalie','Batch','rosaliebatch@hotmail.com','23235514'),
31     ('000000006','Adrian','Palch','arthurbosq@gmail.com','27459121'),
32     ('000000007','Bianca','Onea','biaonea@hotmail.com','44789012'),
33     ('000000008','Sidse','Snomsen','sidses@yahoo.com','57801273'),
34     ('000000009','Yasmin','Fosquetti','slayfosquetti@gmail.com','33890923'),
35     ('000000010','Carina','Nijolek','carinanijo@hotmail.com','29461073'),
36     ('000000011','Paul','McCartney','pmccartney@hotmail.com','38399922'),
37     ('000000012','Snoop','Dogg','snoopydoggy@private.com','77832430'),
38     ('000000013','Jonas','Knocksen','pmccartney@hotmail.com','83901322'),
39     ('000000014','Marilyn','Monroe','marimonroe1@gmail.com','33997283'),
40     ('000000015','Neil','Armstrong','nielastro1@yahoo.com','66249129'),
41     ('000000016','John','Doe','johndoe123@hotmail.com','79823475'),
42     ('000000017','Jane','Door','door@hotmail.com', '79866475'),
43     ('000000018','John','Smith','smithisthebest@hotmail.com','79866575' ),
44     ('000000019','Jane','Yuyu','janethejohn@hotmail.com', '79866475'),
45     ('000000020','Yuri','Da Silva','yudas@gmail.com', '12312345'),
46     ('000000021','Twoja','Stara','tokopara@gmail.com', '21376942'),
47     ('000000022','Syn','Kolezanki','twojejmamy@gmail.com', '2856301'),
48     ('000000023','Tonald','Drump','whitehouse@gmail.com', '66699966');
49
50 # Place inserts
51 insert into Place values
52     # PlacesID, TerminalID, Service
53     ('P001', '1', 'Duty-Free'), ('P002', '1', 'Check-In'),
54     ('P003', '1', 'Check-Out'), ('P004', '1', 'Food'),
55     ('P005', '1', 'Clothes'), ('P006', '2', 'Duty-Free'),
56     ('P007', '2', 'Clothes'), ('P008', '2', 'Check-In'),
57     ('P009', '2', 'Check-Out'), ('P010', '2', 'Books'),
58     ('P011', '3', 'Duty-Free'), ('P012', '3', 'Perfume'),
59     ('P013', '2', 'Clothes'), ('P014', '3', 'Check-In'),
60     ('P015', '3', 'Check-Out'), ('P016', '3', 'Gifts'),
61     ('P017', '4', 'Duty-Free'), ('P018', '4', 'News'),
62     ('P019', '4', 'Check-In'), ('P020', '4', 'Check-Out'),
63     ('P021', '5', 'Check-In'), ('P022', '5', 'Check-Out'),
64     ('P023', '5', 'Duty-Free'), ('P024', '5', 'Food');
65
66
67 insert into Activity values
68     # ActivityID, Service, ActivityDescription, PlaceID, PassportID
69     ('A001', 'Check-In', 'Checked in for flight', 'P002', '000000001'),
70     ('A002', 'Food', 'Bought a burger', 'P004', '000000002'),
71     ('A003', 'Clothes', 'Bought a shirt', 'P005', '000000003'),
72     ('A004', 'Check-Out', 'Checked out of the airport', 'P003', '000000004'),
73     ('A005', 'Duty-Free', 'Bought a bottle of whiskey', 'P001', '000000005'),
74     ('A006', 'Check-In', 'Checked in for flight', 'P008', '000000006'),
75     ('A007', 'Books', 'Bought a book', 'P010', '000000007'),
76     ('A008', 'Clothes', 'Bought on a pair of jeans', 'P007', '000000008'),
77     ('A009', 'Check-Out', 'Checked out of the airport', 'P009', '000000009'),
78     ('A010', 'Gifts', 'Bought a souvenir', 'P016', '000000010'),
79     ('A011', 'Duty-Free', 'Bought a watch', 'P011', '000000011'),
80     ('A012', 'Clothes', 'Bought on a shirt', 'P013', '000000012'),
81     ('A013', 'Check-In', 'Checked in for the flight', 'P008', '000000013'),
82     ('A014', 'Perfume', 'Bought a bottle of perfume', 'P012', '000000014'),
83     ('A015', 'Food', 'Ordered a pizza', 'P004', '000000015');
84
85 # Flights inserts
86 insert into Flight values
87     # FlightID, DepTime, ArrTime, Aircraft, Airline, PassengerCapacity, LuggageCapacity, Sou
```

```
88      ("LAT2359", "2023-04-22 10:50:00.00",  "2023-04-23 19:30:00.00", "Boeing 737", "Latam",
89      ("LAT6893", "2023-04-22 19:30:00.00",  "2023-04-23 10:34:00.00", "Boeing 767", "Latam",
90      ("MAX1234", "2023-04-22 12:00:00.00",  "2023-04-22 20:35:00.00", "Boeing 777", "Maxiflig
91      ("SAS9921", "2023-04-22 11:45:00.00",  "2023-04-22 12:30:00.00", "Boeing 767", "Scandina
92      ("RYN5032", "2023-04-22 11:45:00.00",  "2023-04-23 10:30:00.00", "Boeing 777", "Rynair",
93      ("GOL5021", "2023-04-22 12:15:00.00",  "2023-04-22 13:35:00.00", "Boeing 767", "Gol", 10
94      ("MAL6666", "2023-04-22 06:05:00.00",  "2023-04-22 08:45:00.00", "Boeing 777", "Malokair
95      ("MAL6692", "2023-04-22 18:30:00.00",  "2023-04-22 19:34:00.00", "Boeing 767", "Malokair
96      ("BAL6821", "2023-04-22 10:10:00.00",  "2023-04-22 13:20:00.00", "Boeing 767", "Ballight
97      ("BAL5326", "2023-04-22 12:20:00.00",  "2023-04-22 15:30:00.00", "Boeing 777", "Ballight
98      ("PLO2305", "2023-04-22 23:25:00.00",  "2023-04-22 02:00:00.00", "Boeing 737", "Plane On
99      ("SNP0913", "2023-03-22 14:10:00.00",  "2023-03-22 23:10:00.00", "Boeing 767", "Snoop Li
100     ("LOT2137", "2023-04-22 12:05:00.00",  "2023-04-22 13:00:00.00", "Airbus A380", "LOT", 5
101     ("LOT9202", "2023-04-22 12:05:00.00",  "2023-04-22 13:00:00.00", "Airbus A320", "LOT", 9
102     ("POT5311", "2023-04-22 15:55:00.00",  "2023-04-22 18:00:00.00", "Boeing 767", "Portugal
103     ("XAM2432", "2023-04-22 14:00:00.00",  "2023-04-22 16:00:00.00", "Boeing 777", "Xam Flig
104     ("PLO0239", "2023-04-22 09:00:00.00",  "2023-04-22 11:40:00.00", "Boeing 737", "Plane On
105     ("AMS1240", "2023-03-22 09:30:00.00",  "2023-03-22 11:00:00.00", "Boeing 767", "Amsterda
106     ("HPE5320", "2023-03-22 16:45:00.00",  "2023-03-22 09:30:00.00", "Boeing 737", "Hippie F
107     ("PLG1245", "2023-04-22 19:40:00.00",  "2023-04-23 05:50:00.00", "Boeing 777", "Pluggin
108
109 # Gate inserts
110 insert into Gate values
111     # GateID, FlightID, AllocationStart, AllocationEnd, FloorLevel, TerminalID
112     ( 'A10', "LAT2359", null, null, 0, '1' ),
113     ( 'A11', null, null, null, 0, '1'),
114     ( 'A33', "SAS9921", null, null, 1, '1'),
115     ( 'B01', "RYN5032", null, null, 0, '2'),
116     ( 'B02', null, null, null, 1, '2'),
117     ( 'C33', "BAL5326", null, null, 0, '5');
118
119
120 # Ticket inserts
121 insert into Ticket values
122     # TicketID, Class, PassengerID, FlightID, ValidDate
123     ('223EU89441637','Member','000000001','LAT2359', '2023-04-22'),
124     ('AUT5541903212','Gold','000000002','LAT6893', '2023-04-22'),
125     ('420OL1722640K','Economy','000000003','MAX1234', '2023-04-22'),
126     ('223EU89441638','Member','000000004','SAS9921', '2023-04-22'),
127     ('AUT5541903213','First class','000000008','RYN5032', '2023-04-22'),
128     ('AUT6729483818','Economy','000000021','GOL5021','2023-04-22'),
129     ('AUT5781398346','Economy','000000022','GOL5021','2023-04-22'),
130     ('AUT2345623584','Member','000000023','GOL5021','2023-04-22'),
131     ('AUT5541903214','First class','000000011','GOL5021','2023-04-22'),
132     ('98022MH1370X1','First class','000000015', 'MAL6666','2023-04-22'),
133     ('4815162342LOS','Economy','000000009','MAL6692','2023-04-22'),
134     ('4815162342LOT','Economy','000000010','BAL6821','2023-04-22'),
135     ('AUT5541903215','Economy','000000014','BAL5326','2023-04-22'),
136     ('223EU89441639','Member','000000017','PLO2305','2023-04-22'),
137     ('420OL1722640I','First class','000000012','SNP0913','2023-04-22'),
138     ('8274SW1277464','Economy','000000016','POT5311','2023-04-22'),
139     ('98022MH1370X2','Gold','000000018','XAM2432','2023-04-22'),
140     ('BV1938466382D','Gold','000000019', 'AMS1240','2023-04-22'),
141     ('420OL1722640J','Economy','000000020', 'HPE5320','2023-04-22');
142
143 # Luggage inserts
144 insert into Luggage values
145     # LuggageID, Weight, Delivered, PassportID, TicketID
146     ('19287364','12.30',false,'000000001','223EU89441637'),
147     ('19287365','18.40',false,'000000001','223EU89441637'),
148     ('19287366','7.20',false, '000000001','223EU89441637'),
149     ('92837234','19.40',false,'000000003','420OL1722640K'),
150     ('16252410','17.30',false,'000000004','223EU89441638'),
151     ('16252411','19.10',false,'000000004','223EU89441638'),
152     ('16252412','5.20',false, '000000004','223EU89441638'),
```

```
153     ('76336251','14.50',false,'000000007','AUT5541903212'),
154     ('23235541','11.20',false,'000000008','AUT5541903213'),
155     ('77362513','19.30',false,'000000015','98022MH1370X1'),
156     ('77362514','10.20',false,'000000015','98022MH1370X1'),
157     ('16372827','13.70',false,'000000009','4815162342LOS'),
158     ('72626221','3.20',false, '000000010','4815162342LOT'),
159     ('98987772','10.20',false,'000000011','AUT5541903214'),
160     ('98987773','17.40',false,'000000014','AUT5541903214'),
161     ('16627123','4.20',false,'000000012','420OL1722640I'),
162     ('16627124','4.20',false,'000000012','420OL1722640I'),
163     ('16627125','4.20',true,'000000012','420OL1722640I'),
164     ('16627126','4.20',false,'000000012','420OL1722640I'),
165     ('87693432','16.60',false,'000000016','8274SW1277464'),
166     ('45378284','14.10',false,'000000022','AUT2345623584');
```