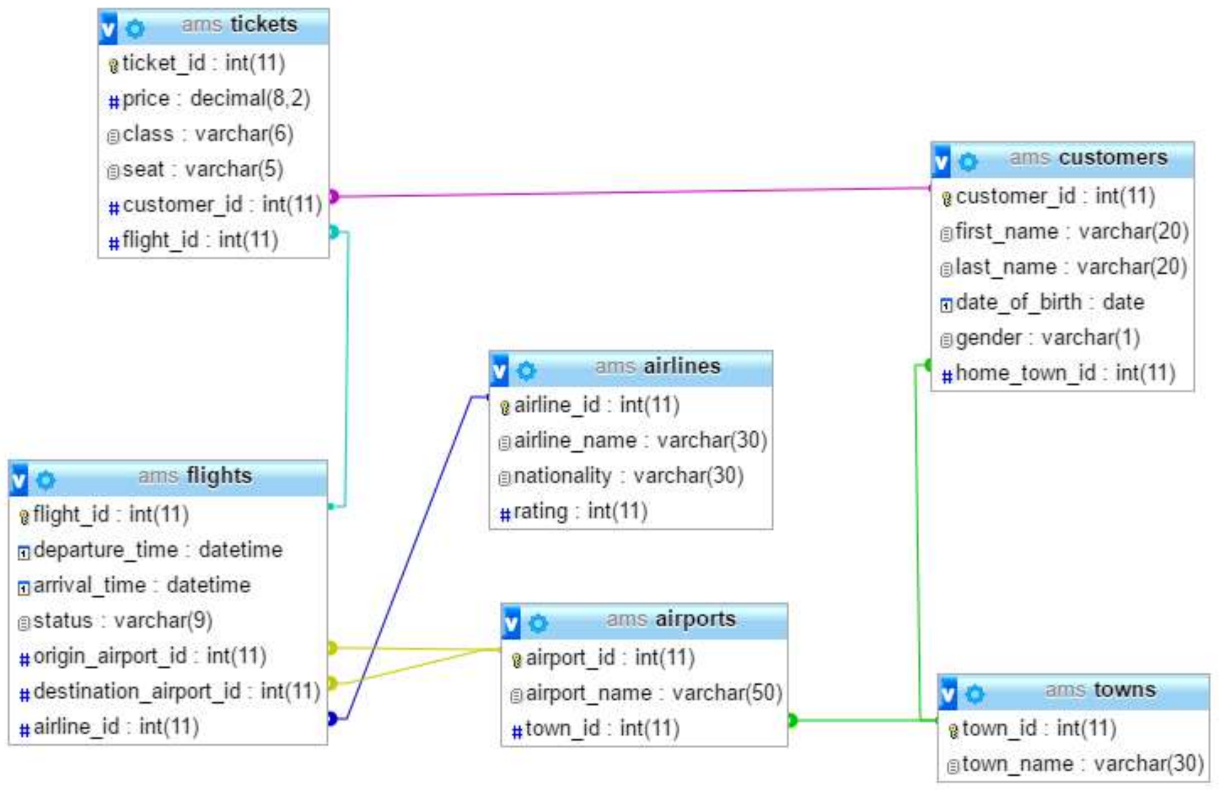
Database Fundamentals MySQL Exam – Airport Management System

You have been assigned to work for the government, on the flight-tracking systems. You've been given access to the AMS Database, which you must modify in several ways in order for you to fulfill your assignment.

Section 0: Database Overview

You have been given an Entity / Relationship Diagram of the AMS Database:



The AMS Database holds information about customers, their tickets, the flights to which the tickets are bought, the flights' origin, destination, airline… And also, the Database holds information about towns.

Make sure you implement the whole database correctly on your local machine, so that you could work with it.

The instructions you'll be given will be the minimal needed for you to implement the database.

Section 1: Data Definition

You will be given 4 of those tables. You need to create tables for **Flights** and **Tickets**. Follow the instructions and constraints about the columns, below, and create the **Tables** exactly as specified.

Submit your queries in judge, using the "**Run Skeleton, Run Queries and Check DB**" execution strategy.

Here is some information about the tables:

towns

Column Name	Data Type	Constraints
town_id	Integer from 1 to 2,147,483,647.	Primary Key
town_name	A string containing a maximum of 30 characters. Unicode is **NOT** needed.	Null is not permitted.

airports

Column Name	Data Type	Constraints
airport_id	Integer from 1 to 2,147,483,647.	Primary Key
airport_name	A string containing a maximum of 50 characters. Unicode is **NOT** needed.	Null is not permitted.
town_id	Integer from 1 to 2,147,483,647.	Relationship with table **towns**.

airlines

Column Name	Data Type	Constraints
airline_id	Integer from 1 to 2,147,483,647.	Primary Key
airline_name	A string containing a maximum of 30 characters. Unicode is **NOT** needed.	Null is not permitted.
nationality	A string containing a maximum of 30 characters. Unicode is **NOT** needed.	Null is not permitted.
rating	Integer from 0 to 2,147,483,647.	Has a **default value** of 0.

customers

Column Name	Data Type	Constraints
customer_id	Integer from 1 to 2,147,483,647	Primary Key
first_name	A string containing a maximum of 20 characters. Unicode is **NOT** needed.	Null is not permitted.
last_name	A string containing a maximum of 20 characters. Unicode is **NOT** needed.	Null is not permitted.
date_of_birth	Date **without** time.	Null is not permitted.
gender	A single character.	Should only contain one of the following values: '**M**', '**F**'.
home_town_id	Integer from 1 to 2,147,483,647	Relationship with table **towns**.

flights

Column Name	Data Type	Constraints
flight_id	Integer from 1 to 2,147,483,647.	Primary Key
departure_time	A Date **with** Time.	Null is not permitted.
arrival_time	A Date **with** Time.	Null is not permitted.
status	A string containing a maximum of 9 characters. Unicode is **NOT** needed.	Should **only contain** one of the following values: ‘**Departing’**, ‘**Delayed**’, ‘**Arrived**’, ‘**Cancelled**’.
origin_airport_id	Integer from 1 to 2,147,483,647.	Relationship with table **airports**.
destination_airport_id	Integer from 1 to 2,147,483,647.	Relationship with table **airports**.
airline_id	Integer from 1 to 2,147,483,647.	Relationship with table **airlines**.

tickets

Column Name	Data Type	Constraints
ticket_id	Integer from 1 to 2,147,483,647.	Primary Key
price	Decimal with length of **8**, 2 digits after the decimal point.	Null is not permitted.
class	A string containing a maximum of 6 characters. Unicode is **NOT** needed.	Should **only contain** one of the following values: ‘**First**’, ‘**Second**’, ‘**Third**’.
seat	A string containing a maximum of 5 characters. Unicode is **NOT** needed.	Null is not permitted.
customer_id	Integer from 1 to 2,147,483,647	Relationship with table **customers**.
flight_id	Integer from 1 to 2,147,483,647	Relationship with table **flights**.

Now that we have our **dummy** database ready, we can start working with it. We know that the main AMS database has a lot of data on it, so we’d want to warm up for it with some data insertions and several data extracts.

Section 2: Database Manipulations

Here we need to do several manipulations in the database, like changing data, adding data, adding tables etc. Submit your queries in judge, using the “**Run Skeleton, Run Queries and Check DB**” execution strategy.

Task 1: Data Insertion

For us to be able to do any manipulations we need some test data first. Insert the following rows of data into their corresponding tables:

flights

flight_id	departure_time	arrival_time	status	origin_airport_id	destination_airport_id	airline_id
1	2016-10-13 06:00 AM	2016-10-13 10:00 AM	Delayed	1	4	1
2	2016-10-12 12:00 PM	2016-10-12 12:01 PM	Departing	1	3	2

3	2016-10-14 03:00 PM	2016-10-20 04:00 AM	Delayed	4	2	4
4	2016-10-12 01:24 PM	2016-10-12 4:31 PM	Departing	3	1	3
5	2016-10-12 08:11 AM	2016-10-12 11:22 PM	Departing	4	1	1
6	1995-06-21 12:30 PM	1995-06-22 08:30 PM	Arrived	2	3	5
7	2016-10-12 11:34 PM	2016-10-13 03:00 AM	Departing	2	4	2
8	2016-11-11 01:00 PM	2016-11-12 10:00 PM	Delayed	4	3	1
9	2015-10-01 12:00 PM	2015-12-01 01:00 AM	Arrived	1	2	1
10	2016-10-12 07:30 PM	2016-10-13 12:30 PM	Departing	2	1	7

tickets

ticket_id	price	class	seat	customer_id	flight_id
1	3000.00	First	233-A	3	8
2	1799.90	Second	123-D	1	1
3	1200.50	Second	12-Z	2	5
4	410.68	Third	45-Q	2	8
5	560.00	Third	201-R	4	6
6	2100.00	Second	13-T	1	9
7	5500.00	First	98-O	2	7

Task 2: Update Arrived Flights

Update all **flights** with **status**-'**Arrived**' **Airline ID**, to **1**.

Task 3: Update Tickets

Find the **highest-rated Airline**, and **increase** all of its **Flights' Tickets' prices** with **50%**.

Task 4: Table Creation

Now we've reached the point where your real assignment comes, and it is to extend the AMS database. The **Customers** would like to write **reviews** about certain **Airlines**. Add to the database a table called **customer_reviews.** Here is what you need to have in it:

customer_reviews

Column Name	Data Type	Constraints
review_id	Integer from 1 to 2,147,483,647.	Primary Key
review_content	A string containing a maximum of **255** characters. Unicode is **NOT** needed.	Null is not permitted.

review_grade	Integer from 0 to 10.	Is in the range (**0 – 10**)
airline_id	Integer from 1 to 2,147,483,647.	Relationship with table **airlines**.
customer_id	Integer from 1 to 2,147,483,647.	Relationship with table **customers**.

The Tickets in the AMS are bought trough bank accounts, and your employers have decided that they need to track that too. You need to add a table that holds data about the Customers' Bank Accounts. Add to the database a table called **customer_bank_accounts.** Here is what you need to have in it:

customer_bank_accounts

Column Name	Data Type	Constraints
account_id	Integer from 1 to 2,147,483,647.	Primary Key
account_number	A string containing a maximum of **10** characters. Unicode is **NOT** needed.	Null is not permitted. Needs to be **unique**.
balance	Decimal with length of **10**, 2 digits after the decimal point.	Null is not permitted.
customer_id	Integer from 1 to 2,147,483,647.	Relationship with table **customers**.

Task 5: Fill the new Tables with Data

Insert the following data into the newly created table – **customer_reviews**.

review_id	review_content	review_grade	airline_id	customer_id
1	Me is very happy. Me likey this airline. Me good.	10	1	1
2	Ja, Ja, Ja… Ja, Gut, Gut, Ja Gut! Sehr Gut!	10	1	4
3	Meh…	5	4	3
4	Well I've seen better, but I've certainly seen a lot worse…	7	3	5

Insert the following data into the newly created table – **customer_bank_accounts.**

account_id	account_number	balance	customer_id
1	123456790	2569.23	1
2	18ABC23672	14004568.23	2
3	F0RG0100N3	19345.20	5

Extract the data in order to see if everything is how it should be.

Note: When you are submitting your results, submit only the insertion queries.

Section 3: Querying

And now we need to do some data extraction. For this section, submit your queries in judge, using the "**Prepare DB and Run Queries**" execution strategy. **Note** that the **example results** from **this section** use a **fresh database**.

Task 1: Extract All Tickets

Extract from the database, all of the Tickets, taking only the **Ticket's ID**, **Price**, **Class** and **Seat**. Sort the results **ascending** by **Ticket ID**.

ticket_id	price	class	seat
1	3000.00	First	233-A
2	1799.90	Second	123-D
3	1200.50	Second	12-Z
4	410.68	Third	45-Q
5	560.00	Third	201-R
6	2100.00	Second	13-T
7	5500.00	First	98-O

Task 2: Extract All Customers

Extract from the database, all of the Customers, taking only the **Customer's ID**, **Full Name** (First name + Last name separated by a **single space**) and **Gender**. Sort the results by **alphabetical order** of the **full name**, and as **second criteria**, sort them **ascending** by **Customer ID**.

customer_id	full_name	gender
1	Cassidy Isacc	F
5	Ivy Indigo	F
2	Jonathan Half	M
4	Joseph Priboi	M
3	Zack Cody	M

Task 3: Extract Delayed Flights

Extract from the database, all of the **Flights**, which have **status**-'**Delayed**', taking only the **Flight's ID**, **Departure Time** and **Arrival Time**. Sort the results **ascending** by **Flight ID**.

flight_id	departure_time	arrival_time
1	2016-10-13 06:00:00.000	2016-10-13 10:00:00.000
3	2016-10-14 15:00:00.000	2016-10-20 04:00:00.000
8	2016-11-11 13:00:00.000	2016-11-12 22:00:00.000

Task 4: Extract Top 5 Most Highly Rated Airlines which have any Flights

Extract from the database, the **top 5** airlines, **in terms of highest rating**, which have **any flights**, taking only the **Airlines' IDs** and **Airlines' Names, Airlines' Nationalities** and **Airlines' Ratings**. If two airlines have **the same rating** order them, **ascending**, by **Airline ID**.

airline_id	airline_name	nationality	rating
1	Royal Airline	Bulgarian	200
2	Russia Airlines	Russian	150
4	Dubai Airlines	Arabian	149
3	USA Airlines	American	100
5	South African Airlines	African	50

Task 5: Extract all Tickets with price below 5000, for First Class

Extract from the database, all tickets, which have **price** below **5000**, and have **class** – '**First**´, taking the **Tickets' IDs**, **Flights' Destination Airport Name**, and **Owning Customers' Full Names** (First name + Last name separated by a **single space**). Order the results, **ascending**, by **Ticket ID**.

ticket_id	destination	customer_name
1	Royals Airport	Zack Cody

Task 6: Extract all Customers which are departing from their Home Town

Extract from the database, all of the **Customers**, which are **departing** from their **Home Town**, taking only the **Customer's ID**, **Full Name** (First name + Last name separated by a **single space**) and **Home Town Name**. Order the results, **ascending**, by **Customer ID**.

customer_id	full_name	home_town
2	Jonathan Half	Moscow

Task 7: Extract all Customers which will fly

Extract from the database all customers, which **have bought tickets** and the **flights** to their **tickets** have **Status**- '**Departing**', taking only the **Customer's ID**, **Full Name** (First name + Last name separated by a **single space**) and **Age**. Order them **Ascending** by their **Age**. Assume that the **current year** is **2016**. If two people have **the same age**, order them by **Customer ID**, **ascending**.

customer_id	full_name	age
2	Jonathan Half	33

Task 8: Extract Top 3 Customers which have Delayed Flights

Extract from the database, the **top 3** Customers, **in terms of most expensive Ticket**, which's flights have **Status**-'**Delayed**'. Take the **Customers' IDs**, **Full Name** (First name + Last name separated by a **single space**), **Ticket Price** and **Flight Destination Airport Name.** If two tickets have **the same price**, order them, **ascending**, by **Customer ID**.

customer_id	full_name	ticket_price	destination
3	Zack Cody	3000.00	Royals Airport
1	Cassidy Isacc	1799.90	Moscow Central Airport
2	Jonathan Half	410.68	Royals Airport

Task 9: Extract the Last 5 Flights, which are departing.

Extract from the database, the **last 5** Flights, **in terms of departure time**, which have a status of '**Departing**', taking only the **Flights' IDs**, **Departure Time**, **Arrival Time, Origin** and **Destination Airport Names**.
You have to take the **last 5 flights in terms of departure time**, which means they must be **ordered ascending by departure time** in the first place. If two flights have the **same departure time**, order them by **Flight ID**, **ascending**.

flight_id	departure_time	arrival_time	origin	destination
5	2016-10-12 08:11:00.000	2016-10-12 23:22:00.000	Moscow Central Airport	Sofia International Airport
2	2016-10-12 12:00:00.000	2016-10-12 12:01:00.000	Sofia International Airport	Royals Airport
4	2016-10-12 13:24:00.000	2016-10-12 16:31:00.000	Royals Airport	Sofia International Airport
10	2016-10-12 19:30:00.000	2016-10-13 12:30:00.000	New York Airport	Sofia International Airport
7	2016-10-12 23:34:00.000	2016-10-13 03:00:00.000	New York Airport	Moscow Central Airport

Task 10: Extract all Customers below 21 years, which have already flew at least once

Extract from the database, all customers which are **below 21 years aged**, and own a **ticket** to a **flight**, which has status – '**Arrived**', taking their **Customer's ID**, **Full Name** (First name + Last name separated by a **single space**), and **Age**. Order them by their **Age** in **descending order**. Assume that the **current year** is **2016**. If two persons have **the same age**, order them by **Customer ID**, **ascending**.

customer_id	full_name	age
1	Cassidy Isacc	19

Task 11: Extract all Airports and the Count of People departing from them

Extract from the database, all airports that have **any flights** with **Status**-'**Departing**', and extract the **count of people that have tickets for those flights**. Take the **Airports' IDs**, **Airports' Names**, and **Count of People** as **'passengers'.** Order the results by **AirportID, ascending.** The flights **must have** some people in them.

airport_id	airport_name	passengers
2	New York Airport	1
4	Moscow Central Airport	1

Section 4: Programmability

Our employers are satisfied with our remarkable skills. They have decided to let us write several stored procedures for the AMS database. For this section, submit your queries in judge, using the "**Run Skeleton, Run Queries and Check DB**" execution strategy.

Task 1: Review Registering Procedure

Write a procedure – "**usp_submit_review**", which registers a review in the **customer_reviews** table. The procedure should accept the following parameters as input:

- customer_id
- review_content
- review_grade
- airline_name

You can assume that the **customer_id** , will always be **valid**, and **existent** in the **database**.

If there is no airline with the given name, raise an error – '**Airline does not exist.**' with **SQL STATE** – '**45000**'.

If no error has been raised, insert the review into the table, with the **Airline's ID**.

Task 2: Ticket Purchase Procedure

Write a procedure – "**usp_purchase_ticket**", which registers a ticket in the **tickets** table, to a customer that has purchased it, taking from his **balance** in the **customer_bank_accounts** table, the provided **ticket price**. The procedure should accept the following parameters as input:

- customer_id
- flight_id
- ticket_price
- class
- seat

You can assume that the **customer_id** , **flight_id**, **class** and **seat** will always be **valid**, and **existent** in the **database**.

If the **ticket price** is **greater** than the **customer's bank account balance**, raise an error '**Insufficient bank account balance for ticket purchase.**', with **SQL STATE** – '**45000**'.

If no error has been raised, **insert** the **ticket** into the table **Tickets**, and **reduce** the **customer's bank account balance** with the **ticket price's value**.

All input parameters will be given in a **valid format**. Numeric data will be given as numbers, text as text etc.

Section 5 (BONUS): Update Trigger

AMS has given you one final task because you are really good. They have already given you full control over their database. You have been tasked to create a table **arrived_flights**, and a **trigger**, which comes in action every time a flight's **status,** is updated to '**Arrived**', and only in that case… In all other cases the update should function normally.

The table should hold basic data about the flight, but also the **amount of passengers**.

The table should have the following columns:

arrived_flights

Column Name	Data Type	Constraints
flight_id	Integer from 1 to 2,147,483,647.	Primary Key
arrival_time	Date with time.	Null is not permitted.
origin	A string containing a maximum of **50** characters. Unicode is **NOT** needed.	Null is not permitted.
destination	A string containing a maximum of **50** characters. Unicode is **NOT** needed.	Null is not permitted.
passengers	Integer from 0 to 2,147,483,647.	Null is not permitted.

If the trigger is triggered, you need to insert the data about the flight in the table **arrived_flights**, but you should also update the flight in the **Flights** table, like it should be done normally.

Submit only the **trigger code**.