# Database Basics (MySQL) Exam Preparation I – 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:

**DIAGRAM HERE**

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.

Your task is to create a database called airport\_management\_system. Then you will have to create several **tables**.

* towns – contains information about the towns.
* airports – contains information about the airports.
  + Each airport has a town.
* airlines – contains geographical information about the location of each person.
* customers – contains information about the chats.
  + Each customer has a home town.
* flights – a mapping table between the users and the chats.
  + Each flight has an origin airport.
  + Each flight has a destination airport.
  + Each flight has an airline.
* tickets – contains information about the messages sent from the users in the chats.
  + Each ticket has a customer.
  + Each ticket has a flight.

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 Language (DDL) – 30 pts

### Table Design

You have been tasked to create the tables in the database by the following models:

**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 **50 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 **0** 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**. | Will contain one of the following values: ‘**M**’, ‘**F**’. |
| home\_town\_id | **Integer**, from **0** to **2,147,483,647.** | Relationship with table **towns**. |

**flights**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| flight\_id | **Integer**, from **0** to **2,147,483,647.** | **Primary Key**  **AUTO\_INCREMENT** |
| departure\_time | Date **WITH** time. | **NULL** is **NOT** permitted. |
| arrival\_time | Date **WITH** time. | **NULL** is **NOT** permitted. |
| status | A string containing a maximum of **9 characters**. Unicode is **NOT** needed. | **WILL** **contain** one of the following values: ‘**Departing’**, ‘**Delayed**’, ‘**Arrived**’, ‘**Cancelled**’. |
| origin\_airport\_id | **Integer**, from **0** to **2,147,483,647.** | Relationship with table airports. |
| destination\_airport\_id | **Integer**, from **0** to **2,147,483,647.** | Relationship with table airports. |
| airline\_id | **Integer**, from **0** to **2,147,483,647.** | Relationship with table airlines. |

**tickets**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| ticket\_id | **Integer**, from **0** to **2,147,483,647.** | **Primary Key**  **AUTO\_INCREMENT** |
| 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. | Will **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 **0** to **2,147,483,647.** | Relationship with table customers. |
| flight\_id | **Integer**, from **0** to **2,147,483,647.** | Relationship with table flights. |

Submit your solutions in Judge on the first task. Submit **all** SQL table creation statements.

## Section 2: Data Manipulation Language (DML) – 30 pts

Here we need to do several manipulations in the database, like changing data, adding data, adding tables etc.

### Data Insertion

You will need to insert several records into the flights table. For every airline with id between **1** and **10 INCLUSIVELY**, insert a record in the flights table.

Depending on several conditions, the **inserted column values** may vary:

* departure\_time – should be set to ‘2017-06-19 14:00:00’.
* arrival\_time – should be set to ‘2017-06-21 11:00:00’.
* status – **DIVIDE** the airline’s id by **4** and take the **REMAINDER**.
  + If the **remainder** is **0**, set the status to ‘Departing’.
  + If the **remainder** is **1**, set the status to ‘Delayed’.
  + If the **remainder** is **2**, set the status to ‘Arrived’.
  + If the **remainder** is **3**, set the status to ‘Canceled’.
* origin\_airport\_id – the **SQUARE ROOT** of the airline’s name’s length.
  + Round the **value UP**.
* destination\_airport\_id – the **SQUARE ROOT** of the airline’s nationality’s length.
  + Round the **value UP**.
* airline\_id – the airline’s id.

### Update Arrived Flights

**Update** all flights with status – ‘Arrived’. Set their airline\_id to **1**.

### Update Tickets

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

## Section 3: Querying – 100 pts

And now we need to do some data extraction. **Note** that the **example results** from **this section** use a **fresh database**.

### Tickets

Extract from the database, all of the tickets, taking only the **ticket’s id**, **price**, **class** and **seat**.   
**ORDER** the results **ascending** by ticket’s id.

|  |  |  |  |
| --- | --- | --- | --- |
| **ticket\_id** | **price** | **class** | **seat** |
| 1 | 100.00 | Third | 55-A |
| ... | ... | ... | ... |

### 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**.

**ORDER** the results in **alphabetical order** of the **full\_name**, and as **second criteria**, order them **ascending** by **customer id**.

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **full\_name** | **gender** |
| 57 | Adam Freeman | M |
| ... | ... | ... |

### Flights

Extract from the database, all of the flights, which have status– ‘**Delayed**’, taking only the **flight’s id**, **departure time** and **arrival time**.

**ORDER** the results **ascending** by **flight id**.

|  |  |  |
| --- | --- | --- |
| **flight\_id** | **departure\_time** | **arrival\_time** |
| 4 | 3000-01-01 14:30:00 | 3000-02-02 12:53:00 |
| ... | ... | ... |

### Top 5 Airlines

Extract from the database, the **TOP 5** airlines, **in terms of HIGHEST** rating, which have **ANY flights**.  
Take only the **airline’s id** and **airline’s name, airline’s nationality** and **airline’s rating**.

**ORDER** the results **ascending** by **airline id**.

|  |  |  |  |
| --- | --- | --- | --- |
| **airline\_id** | **airline\_name** | **nationality** | **rating** |
| 3 | Forca Barca | Spanish | 200 |
| ... | ... | ... | ... |

### ‘First Class’ Tickets

Extract from the database, all tickets, which have price below **5000**, and have class – ‘**First**´, the customers that own them and the flights to which they are.  
Take only the **ticket’s id**, **flight’s destination airport name**, and **customer’s full name**.

**ORDER** the results **ascending**, by **ticket id**.

|  |  |  |
| --- | --- | --- |
| **ticket\_id** | **destination** | **customer\_name** |
| 10 | Montana Broken Airport | Mister Bean |
| ... | ... | ... |

### Home Town Customers

Extract from the database, all of the customers, which are **DEPARTING** from their **home town**, taking only the **customer’s id**, **full name** and **home town name**. Duplicates are non-needed.

**ORDER** the results **ascending**, by **customer id**.

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **full\_name** | **home\_town** |
| 5 | Ivan Ivanov | Barcelona |
| ... | ... | ... |

### Flying Customers

Extract from the database all customers, which **have** tickets, and the flights to their tickets have   
status– ‘**Departing**’. Duplicates are non-needed.  
Take only the **customer’s id**, **full name** and **age**.

**ORDER** the results **ascending** by their **age** and **ascending** by **customer id**, as second criteria.

Assume that the **current year** is **2016**.

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **full\_name** | **age** |
| 21 | Ignatiika Ignatieva | 17 |
| ... | ... | ... |

### Delayed Customers

Extract from the database, the **TOP 3** customers, in terms of **MOST EXPENSIVE ticket**, which’s flights have   
status – ‘**Delayed**’.

Take only the **customer’s id**, **full name**, **ticket price** and **flight destination airport name.**

**ORDER** the results **ascending**, by **customer id**.

|  |  |  |  |
| --- | --- | --- | --- |
| **customer\_id** | **full\_name** | **ticket\_price** | **destination** |
| 57 | Adam Freeman | 8296.00 | Vienna Port |
| ... | ... | ... | ... |

### Last Departing Flights

Extract from the database, the **LAST 5** flights, which will **depart** (have a status – ‘**Departing**’).   
Take only the **flight’s id**, **departure time**, **arrival time, origin** and **destination airport names**.

**ORDER** the results **ascending** by **departure time**, and **ascending** by **flight id** as second criteria.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **flight\_id** | **departure\_time** | **arrival\_time** | **origin** | **destination** |
| 5 | 2016-10-10 00:30:00 | 2016-10-10 06:30:00 | Messi has his own Airport | Istanbul Central Airport |
| ... | ... | ... | ... | ... |

### Flying Children

Extract from the database, all customers which are **aged** **below** **21** **years**, and own a ticket to a flight, which has status – ‘**Arrived**’. Duplicates are non-needed.

Take only the **customer’s id**, **full name**, and **age**.

**ORDER** the results **descending**, by their **age**, and **ascending** by **customer id**, as second criteria.

Assume that the **current year** is **2016**.

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **full\_name** | **age** |
| 22 | George Seatle | 19 |
| ... | ... | ... |

### Airports and Passengers

Extract from the database, all airports that have **ANY flights** with status– ‘**Departing**’, and extract the **COUNT** ofcustomersthat **OWN** ticketsfor thoseflights.   
Take the **airport’s id**, **airport’s name**, and **count of customers** as **‘passengers’.**

**ORDER** the results **ascending** by **airport id.**

**IGNORE** flights with **NO** passengers.

|  |  |  |
| --- | --- | --- |
| **airport\_id** | **airport\_name** | **passengers** |
| 3 | Moscow Central Airport | 2 |
| … | … | … |

## Section 4: Programmability

Your employers are satisfied with your remarkable skills. They have decided to let you write several stored procedures for the AMS database.

For this task you will be introduced to 2 new tables – customer\_reviews and customer\_bank\_accounts. You will be given the ‘**CREATE**’ statements for these 2 tables. You will then have to use them in order to implement the following tasks.

The tables have the following structure:

**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 **1** to **2,147,483,647**. | **WILL ALWAYS** be in 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. |

**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**.**  **Unique** values. |
| 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. |

**NOTE**: The **procedures’ parameters** should be of the same **TYPE** that the **corresponding tables require**.

### Submit Review

Write a procedure – “**udp\_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**.

### Purchase Ticket

Write a procedure – “**udp\_purchase\_ticket**”, which registers a ticket in the ticketstable, 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, classandseat 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 **trigger**, which comes in **action** every time an **UPCOMMING** (a flight that **WILL** fly) **flight’s status**, is **UPDATED** to ‘**Arrived**’, and **ONLY** in that case… In all other cases the update should function normally.

You have been given an extra table – arrived\_flights. The ‘**CREATE**’ statement for it has been given to you. The table has the following structure:

**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 **1** to **2,147,483,647.** | **NULL** is **NOT** permitted**.** |

If the trigger is triggered, you need to **insert** **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**.