# 7BUIS010W: Data Warehousing & Business Intelligence

# Tutorial 2: Review of Data Modelling Principles

Task A:

**Consider the Relational schema of the Flights Database/Source, consisting of 5 five tables (Flights, Flight\_ Instances, Airports, Tickets, Check-in ).**

**Primary Keys are underlined.**

**Foreign Key are denoted as Attribute name: Table Name, i.e. fromAirport: AIRPORTS**,

FLIGHTS(**flight-Number**, airline, **From-Airport: AIRPORTS**, **To-Airport: AIRPORTS,** departure Time, arrivalTime, carrier)

FLIGHT\_INSTANCES (**Flight-Number: FLIGHTS,** **date**)

AIRPORTS (**IATA-Code**, name, city, country)

TICKETS (**Ticket-Number**, **FlightNumber: FLIGHT**\_**INSTANCES,** fare, passengersFirstName, passengersSurname, passengersGender)

CHECK-IN (**Ticket-Number: TICKETS**, **Check-in-Time**, Seat, numberOfBags)

## Simple Questions

1. **What Kind of Business Information each table represents?**
2. **What is the purpose of a Primary Key?**
3. **What is the purpose of a Foreign Key?**

## Further Tasks

1. **What are the main entities or Classes in the case of Flights Database?**
2. **What are the main Relationships and how do you identify them in the case of Flights Database?**
3. **Construct an Entity Relationship model for the case of Flights Database using** [**https://app.diagrams.net/**](https://app.diagrams.net/)
4. **Two different ER Models were constructed model for the case of Flights Database from different members of the Data Modelling Team, see below the proposed ER models. Can be both valid?**
5. **Which of the above tables/Entities might be representing interesting Business Facts/Processes for the purposes of data analysis?**

Diagram

Description automatically generated

Figure ER Models for Flights Databases

Task B:

**Integrate Data-Tables from MYSQL AND SQLITE using Google Colab**

1. **Connect to MYSQL via Python**

Let’s start by installing the necessary packages: install the mysql-connector-python package, a driver to connect Python with MySQL databases

!pip install mysql-connector-python

1. **Setting Up MySQL on Google Colab**

Although Google Colab doesn’t have MySQL pre-installed, we can set it up with a few commands: This installs the MySQL server on our Colab environment.

!apt-get -y install mysql-server

1. **Starting the MySQL Server**

With MySQL installed, let’s start the server:

!service mysql start

You should see a message indicating that MySQL has started.

1. **Secure Your MySQL Installation (Optional)**

Running the MySQL secure installation is usually a good practice, but it involves interactive prompts. In Colab, we’ll execute necessary commands directly:

!mysql -e "ALTER USER 'root'@'localhost' IDENTIFIED WITH 'mysql\_native\_password' BY 'root';FLUSH PRIVILEGES;"

Here, we’re setting the root password as ‘root’. Ensure you use stronger passwords in real-world scenarios!

1. **Connect Securely to MySQL**

Let’s create a connection to our MySQL server using Python:

import mysql.connector

# Create a connection to the MySQL server

conn1 = mysql.connector.connect(user='root', password='root', host='localhost')

# Create a cursor to interact with the MySQL server

cursor = conn1.cursor()

1. **Lets Put it all together through a simple example in MYSQL and SQLITE**

# Create a new database space named 'library1'

cursor.execute("CREATE DATABASE IF NOT EXISTS library1")

# Switch to the 'library' database space

cursor.execute("USE library1")

#create table in database

cursor.execute('''CREATE TABLE jobs1( job\_id INT(3),

                   job\_title VARCHAR(50) unique not null,

                   min\_salary DECIMAL(8,2) not null,

                   max\_salary DECIMAL(8,2) not null,

                   PRIMARY KEY (job\_id));''')

#commit and save changes to database

conn1.commit()

conn1.close()

#Reopen the MYDQL Database

import mysql.connector

# Connect to the MySQL server and the 'library1' database

conn = mysql.connector.connect(user='root', password='root', host='localhost', database='library1')

cursor = conn.cursor()

# Insert sample data on table job in SQLITE

Jobs=[(1901, 'Managing Director', 75000, 125000),

(1902, 'Programmer', 35000, 80000),

(1903, 'Sales Rep', 25000, 45000),

(1904, 'Project Manager', 45000, 95000),

(1905, 'Marketing Manager', 37000, 68000),

(1906, 'Marketing Producer', 34000, 70000),

(1907, 'Operations Manager', 28000, 41000),

(1908, 'Sales Administrator', 38000, 65000),

(1909, 'Database Architect', 44000, 73000),

(1910, 'Operations Officer', 32000, 61000),

(1911, 'IT Tester', 38000, 55000)]

cursor.executemany('''Insert Into jobs1 (job\_id, job\_title, min\_salary, max\_salary) VALUES (%s, %s, %s, %s)''', Jobs)

# Run a Sample Query to SQLITE Server to fetch all records

import pandas as pd

query1 = "SELECT \* FROM jobs1"

df1 = pd.read\_sql\_query(query1,conn)

df1

| **index** | **job\_id** | **job\_title** | **min\_salary** | **max\_salary** |
| --- | --- | --- | --- | --- |
| **0** | 1901 | Managing Director | 75000.0 | 125000.0 |
| **1** | 1902 | Programmer | 35000.0 | 80000.0 |
| **2** | 1903 | Sales Rep | 25000.0 | 45000.0 |
| **3** | 1904 | Project Manager | 45000.0 | 95000.0 |
| **4** | 1905 | Marketing Manager | 37000.0 | 68000.0 |
| **5** | 1906 | Marketing Producer | 34000.0 | 70000.0 |
| **6** | 1907 | Operations Manager | 28000.0 | 41000.0 |
| **7** | 1908 | Sales Administrator | 38000.0 | 65000.0 |
| **8** | 1909 | Database Architect | 44000.0 | 73000.0 |
| **9** | 1910 | Operations Officer | 32000.0 | 61000.0 |
| **10** | 1911 | IT Tester | 38000.0 | 55000.0 |

# create table jobs in SQLITE

import pandas as pd

import sqlite3

# connect to existing database

conn2 = sqlite3.connect('Dem\_table.db')

cur = conn2.cursor()

#create table in database

cur.execute('''CREATE TABLE jobs

                ( job\_id      INT(3),

                 job\_title    VARCHAR(50) unique not null,

                min\_salary    DECIMAL(8,2) not null,

                max\_salary    DECIMAL(8,2) not null,

                PRIMARY KEY (job\_id));

                ''')

#commit and save changes to database

conn2.commit()

# Insert sample data on table job in SQLITE

Jobs=[(901, 'Managing Director', 75000, 125000),

(902, 'Programmer', 35000, 80000),

(903, 'Sales Rep', 25000, 45000),

(904, 'Project Manager', 45000, 95000),

(905, 'Marketing Manager', 37000, 68000),

(906, 'Marketing Producer', 34000, 70000),

(907, 'Operations Manager', 28000, 41000),

(908, 'Sales Administrator', 38000, 65000),

(909, 'Database Architect', 44000, 73000),

(910, 'Operations Officer', 32000, 61000),

(911, 'IT Tester', 38000, 55000)]

cur.executemany('Insert Into jobs Values (?,?,?,?)', Jobs)

# Run a Sample Query to SQLITE Server to fetch all records

query1 = "SELECT \* FROM jobs"

df2 = pd.read\_sql\_query(query1,conn2)

df2

| **index** | **job\_id** | **job\_title** | **min\_salary** | **max\_salary** |
| --- | --- | --- | --- | --- |
| **0** | 1901 | Managing Director | 75000.0 | 125000.0 |
| **1** | 1902 | Programmer | 35000.0 | 80000.0 |
| **2** | 1903 | Sales Rep | 25000.0 | 45000.0 |
| **3** | 1904 | Project Manager | 45000.0 | 95000.0 |
| **4** | 1905 | Marketing Manager | 37000.0 | 68000.0 |
| **5** | 1906 | Marketing Producer | 34000.0 | 70000.0 |
| **6** | 1907 | Operations Manager | 28000.0 | 41000.0 |
| **7** | 1908 | Sales Administrator | 38000.0 | 65000.0 |
| **8** | 1909 | Database Architect | 44000.0 | 73000.0 |
| **9** | 1910 | Operations Officer | 32000.0 | 61000.0 |
| **10** | 1911 | IT Tester | 38000.0 | 55000.0 |

# Merge the jobs table data from SQLITE stored in df2 with jobs table data from MySQL stored in df1

# Stored the merged data in the result pandas frame

result = pd.concat([df1, df2], ignore\_index=True, sort=False)

result

|  | **job\_id** |  | **job\_title** | **min\_salary** | **max\_salary** |
| --- | --- | --- | --- | --- | --- |
| **0** | 901 |  | Managing Director | 75000.0 | 125000.0 |
| **1** | 902 |  | Programmer | 35000.0 | 80000.0 |
| **2** | 903 |  | Sales Rep | 25000.0 | 45000.0 |
| **3** | 904 |  | Project Manager | 45000.0 | 95000.0 |
| **4** | 905 |  | Marketing Manager | 37000.0 | 68000.0 |
| **5** | 906 |  | Marketing Producer | 34000.0 | 70000.0 |
| **6** | 907 |  | Operations Manager | 28000.0 | 41000.0 |
| **7** | 908 |  | Sales Administrator | 38000.0 | 65000.0 |
| **8** | 909 |  | Database Architect | 44000.0 | 73000.0 |
| **9** | 910 |  | Operations Officer | 32000.0 | 61000.0 |
| **10** | 911 |  | IT Tester | 38000.0 | 55000.0 |
| **11** | 912 |  | Finance Director | 72000.0 | 115000.0 |
| **12** | 1901 |  | Managing Director | 75000.0 | 125000.0 |
| **13** | 1902 |  | Programmer | 35000.0 | 80000.0 |
| **14** | 1903 |  | Sales Rep | 25000.0 | 45000.0 |
| **15** | 1904 |  | Project Manager | 45000.0 | 95000.0 |
| **16** | 1905 |  | Marketing Manager | 37000.0 | 68000.0 |
| **17** | 1906 |  | Marketing Producer | 34000.0 | 70000.0 |
| **18** | 1907 |  | Operations Manager | 28000.0 | 41000.0 |
| **19** | 1908 |  | Sales Administrator | 38000.0 | 65000.0 |
| **20** | 1909 |  | Database Architect | 44000.0 | 73000.0 |
| **21** | 1910 |  | Operations Officer | 32000.0 | 61000.0 |
| **22** | 1911 |  | IT Tester | 38000.0 | 55000.0 |

# write the merged data as table named unionoftables in SQLITE

result.to\_sql('unionoftables', con=conn2)

# Retrieve the data  from the table named unionoftables

query3 = "SELECT \* FROM unionoftables"

df3 = pd.read\_sql\_query(query3,conn2)

df3