**LAB02: Business Requirements and Database Design**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submission**:

* Submit a lab file named “int191\_lab02\_xxxxxxxxxxx.docx/.pdf” into the LEB2 system. xxxxxxxxxxx = your student id

**Due Date & Time**:

* Lecturer will inform the LAB02 due date and time in a lab class.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Classicmodels Database - Business Requirements**

**1. Scope**

The Classicmodels database is designed to support the operations of a model car and motorcycle

retailer, covering order management, product inventory, customer information, and employee

assignments.

It aims to store, manage, and retrieve structured data related to:

* Products: details such as product name, scale, vendor, description, stock quantity, and price.
* Customers: contact information, address, and sales representative linkage.
* Orders: order details, status, and related products.
* Employees: information about sales representatives and reporting hierarchy.
* Payments: customer payment records and transaction details.

**2. Features**

The database provides the following core features:

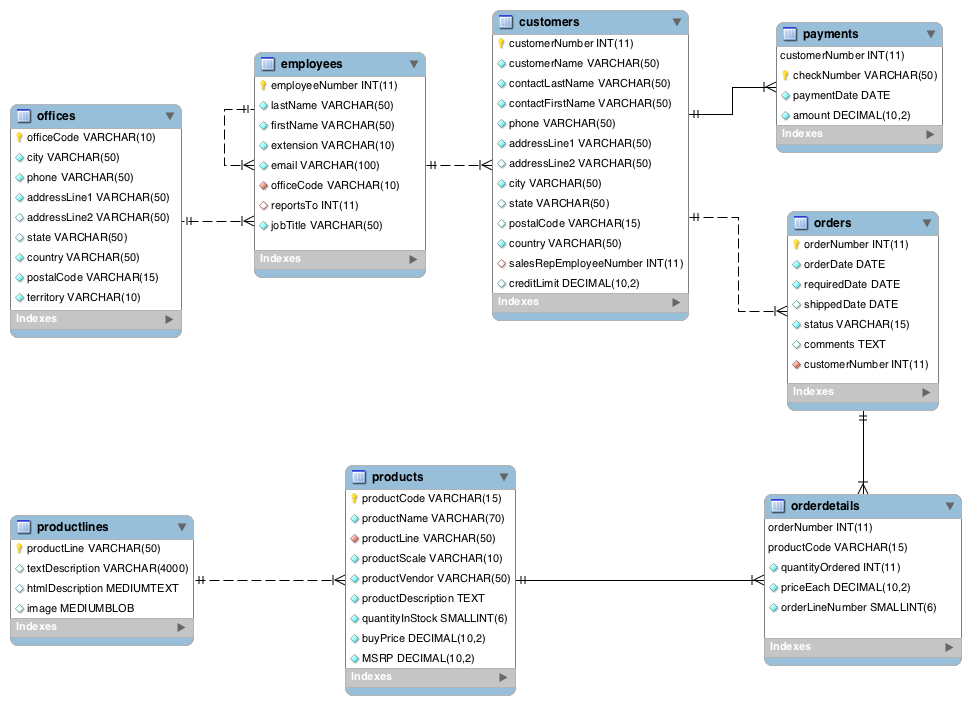
* Customer Management: Store and retrieve customer profiles, contact information, and assigned sales representatives.
* Product Catalog: Maintain a structured list of available products, their specifications, stock levels, and prices.
* Order Tracking: Record orders, including order date, required date, shipped date, and status.
* Payment Recording: Track customer payments, payment dates, amounts, and methods.
* Employee Management: Store employee details, job titles, and reporting structure.
* Reporting Support: Enable queries for sales performance, inventory levels, and customer purchase history.

**3. Users**

The Classicmodels database serves multiple user groups within the organization:

* Sales Representatives: Access customer and order data to follow up on sales leads and manage client relationships.
* Inventory Managers: Monitor stock levels and product availability.
* Finance Staff: Process and verify payment records, track outstanding invoices.
* Management Executives: Retrieve reports for strategic decision-making regarding sales, inventory, and workforce performance.
* Customer Service Staff: Handle customer inquiries and after-sales support.

ER diagram for the **classicmodels** database:

Note: The MSRP is “Manufacturer's suggested retail price” (ราคาขายปลีกที่แนะนำโดยผู้ผลิต).

**SQL - CREATE DATABASE Statement**

CREATE DATABASE creates a database with the given name. To use this statement, you need the CREATE privilege for the database. CREATE SCHEMA is a synonym for CREATE DATABASE.

An error occurs if the database exists and you did not specify IF NOT EXISTS. CREATE DATABASE is not permitted within a session that has an active LOCK TABLES statement.

**CREATE** **{DATABASE | SCHEMA} [IF NOT EXISTS]** db\_name *;*

[create\_option] ...

create\_option: **[DEFAULT]** {

**CHARACTER SET** [=] charset\_name

| **COLLATE** [=] collation\_name

| **ENCRYPTION** [=] {'Y' | 'N'}

}

**SQL - CREATE TABLE Statement**

## SQL provides the CREATE TABLE statement to create a new table in a given database. An SQL query to create a table must define the structure of a table. The structure consists of the name of a table and names of columns in the table with each column's data type. Note that each table must be uniquely named in a database.

**CREATE TABLE** table\_name(

column1 **datatype** [**PRIMARY KEY| UNIQUE| NOT NULL**],

column2 **datatype,**

column3 **datatype,**

.....

columnN **datatype,**

[**PRIMARY KEY**( one or more columns ),

**UNIQUE** (column),

**CHECK** (condition),

**FOREIGN KEY**(column) **REFERENCES** parent\_table\_name (pk\_column)

[**ON DELETE** *reference\_option*]

[**ON UPDATE** *reference\_option*]]

);

reference\_option:

**RESTRICT | CASCADE | SET NULL**

**SQL − ALTER TABLE Statement**

## The SQL ALTER TABLE command is a part of Data Definition Language (DDL) and modifies the structure of a table. The ALTER TABLE command can add or delete columns, create or destroy indexes, change the type of existing columns, or rename columns or the table itself.

**ADD COLUMN:**

**ALTER TABLE** table\_name **ADD** column\_name datatype;

**DROP COLUMN:**

**ALTER TABLE** table\_name **DROP COLUMN** column\_name;

**ADD PRIMARY KEY**

**ALTER TABLE** table\_name

**ADD** [**CONSTRAINT** constraint\_name] **PRIMARY KEY** (column1, column2...);

**DROP PRIMARY KEY**

**ALTER TABLE** table\_name **DROP PRIMARY KEY**;

**ADD CONSTRAINT**

**ALTER TABLE** table\_name

**ADD** [**CONSTRAINT** constraint\_name] **CONSTRAINT\_TYPE**(column1, column2...);

Hint: CONSRAINT\_TYPE: UNIQUE, CHECK, FOREIGN KEY

**DROP CONSTRAINT**

**ALTER TABLE** table\_name **DROP CONSTRAINT** constraint\_name;

Reference: <https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html#foreign-key-examples>

**SQL − DROP TABLE Statement**

DROP TABLE removes one or more tables. You must have the DROP privilege for each table. *Be careful* with this statement! For each table, it removes the table definition and all table data. Dropping a table also drops any triggers for the table.

**DROP TABLE** [IF EXISTS] *tbl\_name* [, *tbl\_name*] ... **;**

**Practice:**

**Task 1:** Evaluate the following business requirements – Student Profile Database

**1. Scope**

The Student Profile Database is designed to store and manage essential academic information for the university. It focuses on maintaining records of students and the academic programs they are studied in, ensuring data accuracy, easy retrieval, and efficient reporting for academic and administrative purposes.

**2. Features**

* Student Information Management: Store personal details such as Student ID, name, date of birth, e-mail, and admission date [วันที่เข้าศึกษา].
* Program Information Management: Maintain program details such as Program ID, program name, program abbreviation [ชื่อย่อหลักสูตร], and faculty [คณะ].
* Admission Linking: Associate each student with their respective academic program.
* Data Integrity: Enforce data consistency through primary and foreign key constraints.

**3. Users**

* Registrar’s Office Staff: For maintaining and updating student and program records.
* Academic Department Staff: For monitoring program admission and student progress.
* University Administrators: For generating reports and supporting decision-making.
  1. List the data that need to be stored for the given requirement in terms of the table.

Example

Table name: XXX

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Constraint | Referenced Table |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Write SQL statements to create a new database named “stdXXX”, XXX is the last three digit of your student ID.

-- Capture the screen of SQL statements and the results and place it here.

* 1. Write SQL statements to create the table(s) in question 1.1 stored in the stdXXX database.
  2. Insert at least three rows into the table(s) and save the data insertion; the data is created by you.