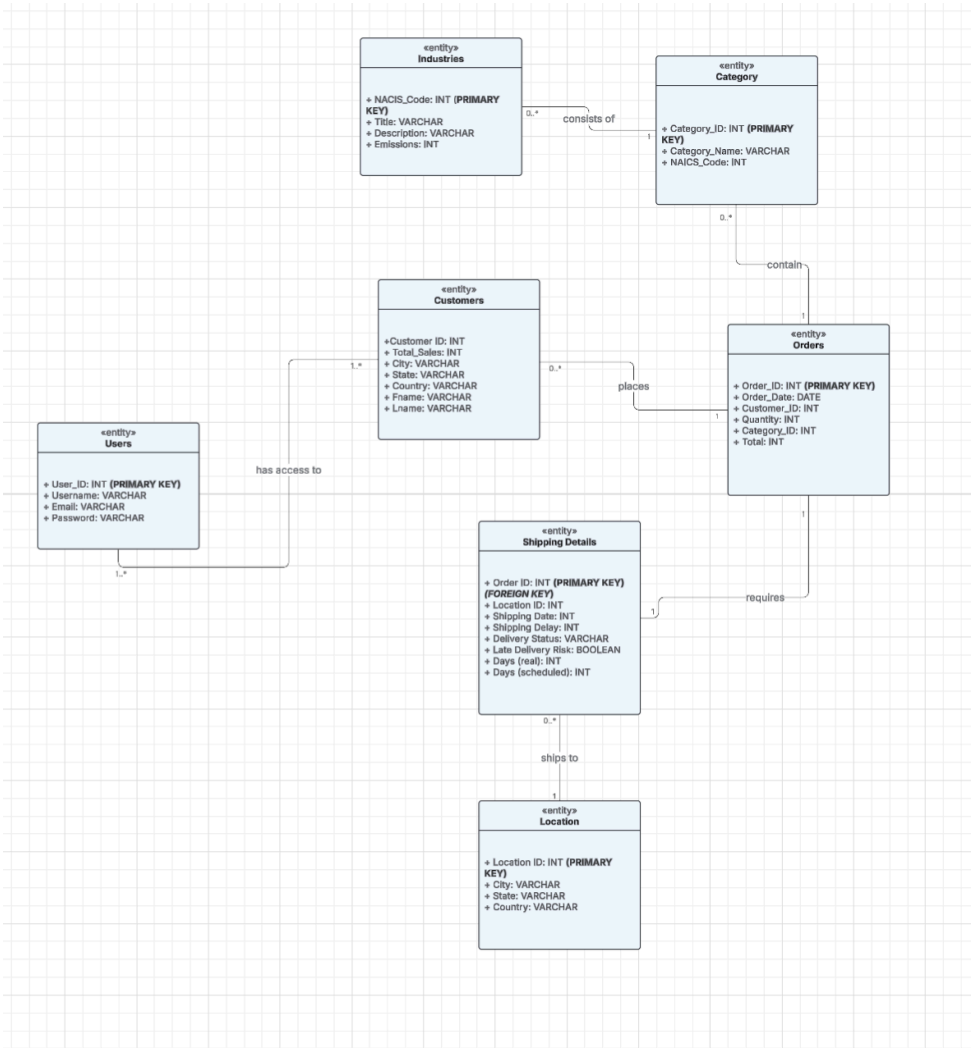
GreenChain Insights
Stage II – Database Design Documentation

1. Introduction

This document presents the **conceptual** and **logical** database design for our project, which aims to track industries and their emission factors, categories associated with each industry, customers and their orders, shipping details for each order, and users (accounts) that manage customer scenarios

---

2. Conceptual Design (UML Diagram)



3. Entities and Assumptions

Below are the entities, their primary keys (PK), attributes, and justifications.

3.1 Industries

- **Primary Key**: `NAICS_Code (INT)`
- **Attributes**: `Title (VARCHAR)`, `Description (VARCHAR)`, `Emission_Factor (DECIMAL)`
- **Assumptions**:
    1. **Multiple Attributes**: Each industry has a title, a detailed description, and an emission factor representing environmental impact, so it is a natural standalone entity rather than just an attribute of something else.
    2. **Standard Codes**: NAICS codes provide a standard classification, making them ideal as a primary key.

3.2 Category

- **Primary Key**: `Category_ID (INT)`
- **Attributes**: `Category_Name (VARCHAR)`, `NAICS_Code (FK → Industries.NAICS_Code)`
- **Assumptions**:
    1. **Dependent on Industry**: Each category belongs to exactly one industry.
    2. **Flexible Schema**: Storing category details separately allows more flexible structuring (e.g., adding category-specific attributes later).

3.3 Customers

- **Primary Key**: `Customer_ID (INT)`
- **Attributes**: `Fname (VARCHAR)`, `Lname (VARCHAR)`, `Total_Sales (DECIMAL)`, `City (VARCHAR)`, `State (VARCHAR)`, `Country (VARCHAR)`
- **Assumptions**:
    1. **Distinct from Orders**: A customer can have many orders, so storing them in a separate table avoids redundancy.
    2. **Analytical Usage**: Fields like City/State/Country allow demographic and geographic analytics.

3.4 Users

- **Primary Key**: `User_ID (INT)`
- **Attributes**: `Username (VARCHAR)`, `Email (VARCHAR)`, `Password (VARCHAR)`
- **Assumptions**:

1. **Security**: Login information is separated for security.
2. **One Entity Only**: At most one entity is allowed for user credentials (as per assignment requirement).

3.5 Orders

- **Primary Key**: `Order_ID (INT)`
- **Attributes**: `Order_Date (DATE)`, `Customer_ID (FK → Customers.Customer_ID)`, `Category_ID (FK → Category.Category_ID)`, `Quantity (INT)`, `Order_Total (DECIMAL)`
- **Assumptions**:
    1. **Transaction Focus**: Each order is a transaction linking a single customer to a category/product line.
    2. **Metric Aggregation**: `Order_Total` is aggregated from other factors (e.g., price × quantity).

3.6 Shipping_Details

- **Primary Key**: `Order_ID (INT) [FK → Orders.Order_ID]`
- **Attributes**: `Location_ID (INT) [FK → Location.Location_ID]`, `Shipping_Date (DATE)`, `Shipping_Delay (INT)`, `Delivery_Status (VARCHAR)`, `Late_Delivery_Risk (BOOLEAN)`, `Days_Real (INT)`, `Days_Scheduled (INT)`
- **Assumptions**:
    1. **1:1 with Orders**: Each order has exactly one shipping detail (PK = `Order_ID`).
    2. **Complex Fields**: Separate shipping details allows for advanced logistics queries (delays, statuses, risk, etc.).

3.7 Location

- **Primary Key**: `Location_ID (INT)`
- **Attributes**: `City (VARCHAR)`, `State (VARCHAR)`, `Zipcode (VARCHAR or INT)`, `Country (VARCHAR)`
- **Assumptions**:
    1. **1:Many with Shipping_Details**: A single location can be referenced by multiple shipping records.
    2. **Reuse**: Normalising location data avoids repeated entries in shipping details.

---

4. Relationship Cardinalities

4.1 Industries → Category (1:M)

- **Justification**: One industry can contain multiple categories (e.g., sub-classifications).
- **Cardinality**: One Industry to Many Categories.

4.2 Category → Orders (1:M)

- **Justification**: A single category (e.g., “Electronics”) can appear in multiple orders.
- **Cardinality**: One Category to Many Orders.

4.3 Customers → Orders (1:M)

- **Justification**: A single customer can place multiple orders, but each order belongs to exactly one customer.
- **Cardinality**: One Customer to Many Orders.

4.4 Orders → Shipping_Details (1:1)

- **Justification**: Each order has exactly one corresponding shipping record.
- **Cardinality**: One Order to One Shipping_Details (implemented via shared primary key).

4.5 Location → Shipping_Details (1:M)

- **Justification**: A single location (address) may be used for multiple shipping instances (though each shipping detail references only one location).
- **Cardinality**: One Location to Many Shipping_Details.

4.6 Users → Customers (M:N)

- **Justification**: A single user may oversee multiple customers, and likewise, multiple users can collaborate on the same customer account. This flexibility better reflects real-world scenarios, where roles and responsibilities can overlap.
- **Cardinality**: Many Users to Many Customers

---

5. Normalization

We applied **Boyce–Codd Normal Form (BCNF)**. Each table has a single candidate key, and all non-key attributes depend on the key, the whole key, and nothing but the key.

5.1 Industries

- **Schema**: `Industries(NAICS_Code [PK], Title, Description, Emission_Factor)`
- **FD**: `NAICS_Code → Title, Description, Emission_Factor`
- **BCNF Check**: `NAICS_Code` is the sole candidate key. No partial or transitive dependencies.

5.2 Category

- **Schema**: `Category(Category_ID [PK], Category_Name, NAICS_Code [FK])`
- **FD**: `Category_ID → Category_Name, NAICS_Code`
- **BCNF Check**: `Category_ID` is the sole candidate key. All attributes depend directly on it.

5.3 Customers

- **Schema**: `Customers(Customer_ID [PK], Fname, Lname, Total_Sales, City, State, Country)`
- **FD**: `Customer_ID → Fname, Lname, Total_Sales, City, State, Country`
- **BCNF Check**: Single candidate key. No extra dependencies.

5.4 Users

- **Schema**: `Users(User_ID [PK], Username, Email, Password)`
- **FD**: `User_ID → Username, Email, Password`
- **BCNF Check**: Single candidate key. No transitive dependencies.

5.5 Orders

- **Schema**: `Orders(Order_ID [PK], Order_Date, Customer_ID [FK], Category_ID [FK], Quantity, Order_Total)`
- **FD**: `Order_ID → Order_Date, Customer_ID, Category_ID, Quantity, Order_Total`
- **BCNF Check**: Single candidate key. All attributes depend on `Order_ID`.

5.6 Shipping_Details

- **Schema**: `Shipping_Details(Order_ID [PK, FK], Location_ID [FK], Shipping_Date, Shipping_Delay, Delivery_Status, Late_Delivery_Risk, Days_Real, Days_Scheduled)`
- **FD**: `Order_ID → (all attributes above)`
- **BCNF Check**: Single candidate key is `Order_ID`. No partial or transitive dependencies.

5.7 Location

- **Schema**: `Location(Location_ID [PK], City, State, Zipcode, Country)`
- **FD**: `Location_ID → City, State, Zipcode, Country`
- **BCNF Check**: Single candidate key, no extra dependencies.

---

6. Logical Design (Relational Schema)

In line with the required format, we list each table, its columns, data types, and PK/FK indicators:

```
Industries(NAICS_Code:INT [PK], Title:VARCHAR(100),
Description:VARCHAR(255), Emission_Factor:DECIMAL(8,2))

Category(Category_ID:INT [PK], Category_Name:VARCHAR(100),
NAICS_Code:INT [FK to Industries.NAICS_Code])

Customers(Customer_ID:INT [PK], Fname:VARCHAR(50), Lname:VARCHAR(50),
Total_Sales:DECIMAL(10,2), City:VARCHAR(50), State:VARCHAR(50),
Country:VARCHAR(50))

Users(User_ID:INT [PK], Username:VARCHAR(50), Email:VARCHAR(100),
Password:VARCHAR(100))

Orders(Order_ID:INT [PK], Order_Date:DATE, Customer_ID:INT [FK to
Customers.Customer_ID], Category_ID:INT [FK to Category.Category_ID],
Quantity:INT, Order_Total:DECIMAL(10,2))

Shipping_Details(Order_ID:INT [PK, FK to Orders.Order_ID],
Location_ID:INT [FK to Location.Location_ID], Shipping_Date:DATE,
Shipping_Delay:INT, Delivery_Status:VARCHAR(50),
Late_Delivery_Risk:BOOLEAN, Days_Real:INT, Days_Scheduled:INT)
```

```
Location(Location_ID:INT [PK], City:VARCHAR(50), State:VARCHAR(50),
Zipcode:VARCHAR(20), Country:VARCHAR(50))
```
