Data Analytics sample

Database schema that you can use for data cleanup and preparation practice as part of a MySQL .

The dataset represents a basic **sales database** containing customer, order, product, and sales data, with common data quality issues such as missing values, duplicates, and inconsistencies.

1. Database Schema Overview

The database consists of 4 tables:

- **customers**: Information about customers.
- **products**: Information about products.
- **orders**: Information about customer orders.
- **sales**: Information about individual product sales within orders.

2. Tables and Sample Data

1. `customers` Table

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone VARCHAR(15),
    city VARCHAR(50),
    state VARCHAR(50),
    country VARCHAR(50)
);

INSERT INTO customers (customer_id, first_name, last_name, em
ail, phone, city, state, country) VALUES
```

```
(1, 'John', 'Doe', 'johndoe@example.com', '1234567890', 'New
York', 'NY', 'USA'),
(2, 'Jane', 'Smith', NULL, '2345678901', 'Los Angeles', 'CA',
'USA'),
(3, 'Alex', 'Johnson', 'alexjohnson@example.com', '345678901
2', 'London', NULL, 'UK'),
(4, 'Emily', 'Brown', 'emilybrown@example.com', '1234567890',
'Los Angeles', 'CA', 'USA'),
(5, 'Chris', 'Doe', 'chrisdoe@example.com', NULL, 'Sydney', N
ULL, 'Australia');
```

Common Data Quality Issues:

- Missing emails for some customers.
- Duplicated phone numbers (e.g., John Doe and Emily Brown have the same phone number).
- Missing `state` values.

---

2. `products` Table

```
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50),
    price DECIMAL(10, 2)
);

INSERT INTO products (product_id, product_name, category, pri
ce) VALUES
(1, 'Laptop', 'Electronics', 1200.00),
(2, 'Smartphone', 'Electronics', 800.00),
(3, 'Desk Chair', 'Furniture', 150.00),
(4, 'Tablet', 'Electronics', NULL),
(5, 'Lamp', 'Furniture', 40.00);
```

Common Data Quality Issues:

- Missing product price for the `Tablet`.
- Some categories might need cleanup (e.g., merging similar categories like `Electronics` with possible future misspellings).

---

3. `orders` Table

```
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    total_amount DECIMAL(10, 2),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_i
d)
);

INSERT INTO orders (order_id, customer_id, order_date, total_
amount) VALUES
(1, 1, '2024-08-01', 1350.00),
(2, 2, '2024-08-03', 840.00),
(3, 3, NULL, 150.00),
(4, 5, '2024-08-05', 40.00),
(5, 1, '2024-08-10', 1200.00);
```

Common Data Quality Issues:

- Missing `order_date` for some orders.
- Possible inconsistency in `total_amount` values based on sales data.

---

4. `sales` Table

```
CREATE TABLE sales (
    sale_id INT PRIMARY KEY,
    order_id INT,
```

```
    product_id INT,
    quantity INT,
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);

INSERT INTO sales (sale_id, order_id, product_id, quantity) V
ALUES
(1, 1, 1, 1),
(2, 1, 3, 1),
(3, 2, 2, 1),
(4, 3, 3, 1),
(5, 4, 5, 1),
(6, 5, 1, 1);
```

Common Data Quality Issues:

- Potential mismatch between the `total_amount` in `orders` and the product prices in `sales`.
- Missing or incorrect `quantity` values that don't align with total order amounts.

---

3. Practice Tasks for Data Cleanup

Here are some cleanup tasks you can practice in your MySQL introductory course:

1. **Identify and remove duplicate data**:
    - Find and remove customers with the same phone numbers.

    ```
    SELECT phone, COUNT(*)
    FROM customers
    GROUP BY phone
    HAVING COUNT(*) > 1;
    ```

2. **Fix missing data**:
    - Fill in the missing `email` and `state` values for customers.

- Set a default price for products with `NULL` values.
- Fix the `order_date` for orders where it's missing.

3. **Data consistency check**:
   - Ensure that the total price in the `orders` table matches the total sales amount for the products in the `sales` table.

```
SELECT o.order_id, o.total_amount, SUM(p.price * s.quantit
y) AS calculated_total
FROM orders o
JOIN sales s ON o.order_id = s.order_id
JOIN products p ON s.product_id = p.product_id
GROUP BY o.order_id
HAVING o.total_amount != calculated_total;
```

4. **Format data**:
   - Standardize customer names to proper case.
   - Ensure that all email addresses are in lowercase.
5. **Analyze relationships**:
   - Create queries to analyze customer purchase behavior (e.g., most purchased products by customer).

---

4. How to Set Up the Database

1. **Create the database**:

```
CREATE DATABASE sales_db;
USE sales_db;
```

2. **Create and populate the tables** using the SQL commands provided above.
3. **Perform the data cleanup tasks** as outlined in the practice tasks section.