1. In an absolutely unprecedented and rather questionable decision, the International Olympic Committee (IOC) assigned the critical task of designing and managing the athlete database for the upcoming Olympic Games to none other than Homer and Bart Simpson. Predictably, their complete lack of expertise in databases resulted in a design that was, to put it mildly, a disaster—a chaotic mess full of redundancies, inconsistencies, and questionable logic.

But the comedy of errors didn't stop there. When it came time to input the data, the responsibility was handed over to Barney Gumble, who, true to form, decided to do the job while in the midst of an epic bender at Moe's Tavern. Unsurprisingly, the result was a database filled with errors, inconsistencies, and general mayhem—like data inserted backward, names spelled in ways that defy the alphabet, and some athletes listed as participating in events that haven't even been invented yet.

To make matters even worse, it turns out that Homer was under intense pressure from his boss, Mr. Burns, to ensure that the United States won every single event. In his panic and confusion—likely exacerbated by a doughnut-induced sugar high—Homer completely forgot to include athletes from the Soviet Union (URSS). This glaring omission has caused an international uproar, with conspiracy theories running wild and accusations flying left and right.

Now, the entire world is looking at you, a budding database expert, to clean up this mess. Your mission is to normalize the database (because let's be honest, it's a hot mess right now), create the necessary normalized tables in PostgreSQL, correctly input the data, and then perform a series of SQL queries to make sure everything is finally in order. This isn't just about fixing a database; it's about restoring the dignity of the Olympics—and maybe, just maybe, ensuring Homer doesn't get fired… again.

So roll up your sleeves, grab your SQL tools, and get ready to show the world (and the Simpsons) how it's done!

In file Olympics.csv you can find the data, please normalize in third normal form. And then develop the following queries:

1. Retrieve all female athletes (gender = 'Female') who participate in more than one discipline. Display their full names, disciplines, and number of disciplines (5).
2. List the full names and nationalities of athletes who were born after the year 2000 and weigh more than the average weight of all athletes (5).
3. Count the number of athletes per country (country_full), and display the result only for countries that have more than five athletes (5).
4. Find the youngest and oldest athletes in the database. Show their full names, birth dates, and nationalities (5).
5. List the athletes who participate in more than three events, displaying their full names, the events, and the total number of events they participate in. Use a temporary table to assist in this query (5).

6. Create a view that lists all athletes along with the number of events they participate in. Then, write a query to display the athletes from this view who participate in exactly two events (5).
7. Use a temporary table to list the average height and weight of athletes grouped by nationality. Then, query this table to find nationalities where the average height is above 180 cm and the average weight is below 75 kg (5).
8. Write a query to identify any potential data inconsistencies by finding athletes whose listed events do not match their disciplines. For example, if an athlete's discipline is 'Archery' but their events include 'Swimming', this should be flagged as inconsistent. Use a combination of joins, string functions, and subqueries to achieve this, and display the athlete's name, discipline, and the mismatched events (5).
9. Database normalized and postgresql script for database and data insertion (10)

2. In an unfortunate turn of events, Daniel Pineda and Pedro Orrego, who were responsible for designing and managing the database of classified apartment rentals, were unceremoniously fired. The reason? A disastrous database structure filled with inconsistencies, redundant data, and an alarming number of errors. To make matters worse, the data was entered haphazardly, leading to corrupted fields and missing information.

As a result, the company has turned to you, a skilled database expert, to salvage what's left of the project. Your mission is to normalize the database, create the necessary tables in PostgreSQL, and perform a series of SQL queries to ensure the data is structured and accessible. This task will not only test your knowledge of database normalization but also your ability to work with complex SQL queries in a real-world scenario. Your task is shown below:

1. Normalize the database up to the Third Normal Form (3NF). Break down the large, unstructured table into smaller, well-structured tables to eliminate redundancy and ensure data integrity (8%).
2. Create the normalized tables in PostgreSQL. Ensure that the relationships between tables are properly defined using foreign keys (1%).
3. Insert the cleaned data into the normalized tables (1%).
4. Retrieve all apartment listings located in a specific city, displaying the title, price, and number of bedrooms (5%).
5. Count the number of listings per state and display states that have more than 100 listings (5%).
6. Find the average price of apartments in each city that have at least two bedrooms. Display the city name, average price, and the number of apartments meeting the criteria (5%).
7. List all apartments that include the word "luxury" in their title or description, along with their price and address (5%).
8. Create a view that lists all apartments along with their price per square foot. Then, write a query to display apartments from this view that have a price per square foot above the median value (5%).

9. Use a temporary table to store the top 10 most expensive apartments in each state. Then, write a query to find the average price of these top listings by state (5%).
10. Identify inconsistencies in the data by finding listings where the stated number of bedrooms does not match the description. For example, if an apartment is listed as having 3 bedrooms but the description mentions only 2, this should be flagged. Use a combination of string functions, joins, and subqueries to accomplish this, and display the listing ID, title, and the inconsistency (5%).