Dataset: Fitness Tracker Data

This dataset contains information from a fitness tracker app. Each row represents a user's daily activity, including steps taken, calories burned, distance traveled, and active minutes.

Sample Data:

```
user_id,date,steps,calories,distance_km,active_minutes
1,2023-07-01,12000,500,8.5,90
2,2023-07-01,8000,350,5.6,60
3,2023-07-01,15000,600,10.2,120
1,2023-07-02,11000,480,7.9,85
2,2023-07-02,9000,400,6.2,70
3,2023-07-02,13000,520,9.0,100
1,2023-07-03,10000,450,7.1,80
2,2023-07-03,7000,320,4.9,55
3,2023-07-03,16000,620,11.0,130
```

Exercises:

1. **Find the Total Steps Taken by Each User**
    - Calculate the total number of steps taken by each user across all days.
2. **Filter Days Where a User Burned More Than 500 Calories**
    - Identify all days where a user burned more than 500 calories.
3. **Calculate the Average Distance Traveled by Each User**
    - Calculate the average distance traveled (`distance_km`) by each user across all days.
4. **Identify the Day with the Maximum Steps for Each User**
    - For each user, find the day when they took the maximum number of steps.
5. **Find Users Who Were Active for More Than 100 Minutes on Any Day**
    - Identify users who had active minutes greater than 100 on any day.
6. **Calculate the Total Calories Burned per Day**
    - Group by `date` and calculate the total number of calories burned by all users combined for each day.
7. **Calculate the Average Steps per Day**
    - Find the average number of steps taken across all users for each day.
8. **Rank Users by Total Distance Traveled**
    - Rank the users by their total distance traveled, from highest to lowest.
9. **Find the Most Active User by Total Active Minutes**
    - Identify the user with the highest total active minutes across all days.
10. **Create a New Column for Calories Burned per Kilometer**

- Add a new column called `calories_per_km` that calculates how many calories were burned per kilometer (`calories / distance_km`) for each row.

Dataset: Book Sales Data

This dataset contains information about book sales in a store. Each row represents a sale, including details about the book, author, genre, sale price, and the date of the transaction.

Sample Data:

```
sale_id,book_title,author,genre,sale_price,quantity,date
1,The Catcher in the Rye,J.D. Salinger,Fiction,15.99,2,2023-01-05
2,To Kill a Mockingbird,Harper Lee,Fiction,18.99,1,2023-01-10
3,Becoming,Michelle Obama,Biography,20.00,3,2023-02-12
4,Sapiens,Yuval Noah Harari,Non-Fiction,22.50,1,2023-02-15
5,Educated,Tara Westover,Biography,17.99,2,2023-03-10
6,The Great Gatsby,F. Scott Fitzgerald,Fiction,10.99,5,2023-03-15
7,Atomic Habits,James Clear,Self-Help,16.99,3,2023-04-01
8,Dune,Frank Herbert,Science Fiction,25.99,1,2023-04-10
9,1984,George Orwell,Fiction,14.99,2,2023-04-12
10,The Power of Habit,Charles Duhigg,Self-Help,18.00,1,2023-05-01
```

Exercises:

1. **Find Total Sales Revenue per Genre**
    - Group the data by `genre` and calculate the total sales revenue for each genre. (Hint: Multiply `sale_price` by `quantity` to get total sales for each book.)
2. **Filter Books Sold in the "Fiction" Genre**
    - Filter the dataset to include only books sold in the "Fiction" genre.
3. **Find the Book with the Highest Sale Price**
    - Identify the book with the highest individual sale price.
4. **Calculate Total Quantity of Books Sold by Author**
    - Group the data by `author` and calculate the total quantity of books sold for each author.
5. **Identify Sales Transactions Worth More Than $50**
    - Filter the sales transactions where the total sales amount (sale_price * quantity) is greater than $50.
6. **Find the Average Sale Price per Genre**
    - Group the data by `genre` and calculate the average sale price for books in each genre.
7. **Count the Number of Unique Authors in the Dataset**

- Count how many unique authors are present in the dataset.

8. **Find the Top 3 Best-Selling Books by Quantity**
    - Identify the top 3 best-selling books based on the total quantity sold.
9. **Calculate Total Sales for Each Month**
    - Group the sales data by month and calculate the total sales revenue for each month.
10. **Create a New Column for Total Sales Amount**
    - Add a new column `total_sales` that calculates the total sales amount for each transaction (`sale_price * quantity`).

Dataset: Food Delivery Orders

This dataset contains information about food delivery orders placed by customers. Each row represents a single order, including details like the **order ID**, **customer ID**, **restaurant name**, **food item**, **quantity**, **price**, **delivery time**, and **order date**.

Sample Data:

```
order_id,customer_id,restaurant_name,food_item,quantity,price,delivery_time_mins,order_d

1,201,McDonald's,Burger,2,5.99,30,2023-06-15
2,202,Pizza Hut,Pizza,1,12.99,45,2023-06-16
3,203,KFC,Fried Chicken,3,8.99,25,2023-06-17
4,201,Subway,Sandwich,2,6.50,20,2023-06-17
5,204,Domino's,Pizza,2,11.99,40,2023-06-18
6,205,Starbucks,Coffee,1,4.50,15,2023-06-18
7,202,KFC,Fried Chicken,1,8.99,25,2023-06-19
8,206,McDonald's,Fries,3,2.99,15,2023-06-19
9,207,Burger King,Burger,1,6.99,30,2023-06-20
10,203,Starbucks,Coffee,2,4.50,20,2023-06-20
```

Exercises:

1. **Calculate Total Revenue per Restaurant**
    - Group the data by `restaurant_name` and calculate the total revenue for each restaurant. (Hint: Multiply `price` by `quantity` to get total revenue per order.)
2. **Find the Fastest Delivery**
    - Identify the order with the fastest delivery time.
3. **Calculate Average Delivery Time per Restaurant**
    - Group the data by `restaurant_name` and calculate the average delivery time for each restaurant.
4. **Filter Orders for a Specific Customer**

- Filter the dataset to include only orders placed by a specific customer (e.g., `customer_id = 201`).

5. **Find Orders Where Total Amount Spent is Greater Than $20**
   - Filter orders where the total amount spent (price * quantity) is greater than $20.
6. **Calculate the Total Quantity of Each Food Item Sold**
   - Group the data by `food_item` and calculate the total quantity of each food item sold.
7. **Find the Top 3 Most Popular Restaurants by Number of Orders**
   - Identify the top 3 restaurants with the highest number of orders placed.
8. **Calculate Total Revenue per Day**
   - Group the data by `order_date` and calculate the total revenue for each day.
9. **Find the Longest Delivery Time for Each Restaurant**
   - For each restaurant, find the longest delivery time.
10. **Create a New Column for Total Order Value**
    - Add a new column `total_order_value` that calculates the total value of each order (`price * quantity`).

Dataset: Weather Data

This dataset contains daily weather observations recorded in different cities. Each row represents the weather data for a specific city on a given day, including the temperature, humidity, wind speed, and the condition of the day.

Sample Data:

```
date,city,temperature_c,humidity,wind_speed_kph,condition
2023-01-01,New York,5,60,20,Cloudy
2023-01-01,Los Angeles,15,40,10,Sunny
2023-01-01,Chicago,-2,75,25,Snow
2023-01-02,New York,3,65,15,Rain
2023-01-02,Los Angeles,18,35,8,Sunny
2023-01-02,Chicago,-5,80,30,Snow
2023-01-03,New York,6,55,22,Sunny
2023-01-03,Los Angeles,20,38,12,Sunny
2023-01-03,Chicago,-1,70,18,Cloudy
```

Exercises:

1. **Find the Average Temperature for Each City**
   - Group the data by `city` and calculate the average temperature for each city.
2. **Filter Days with Temperature Below Freezing**

- Filter the data to show only the days where the temperature was below freezing (below 0°C).

3. **Find the City with the Highest Wind Speed on a Specific Day**
   - Find the city with the highest wind speed on a specific day (e.g., `2023-01-02`).
4. **Calculate the Total Number of Days with Rainy Weather**
   - Count the number of days where the `condition` was "Rain."
5. **Calculate the Average Humidity for Each Weather Condition**
   - Group the data by `condition` and calculate the average humidity for each weather condition (e.g., Sunny, Rainy, Cloudy).
6. **Find the Hottest Day in Each City**
   - For each city, find the day with the highest recorded temperature.
7. **Identify Cities That Experienced Snow**
   - Filter the dataset to show only the cities that experienced "Snow" in the `condition`.
8. **Calculate the Average Wind Speed for Days When the Condition was Sunny**
   - Filter the dataset for `condition = 'Sunny'` and calculate the average wind speed on sunny days.
9. **Find the Coldest Day Across All Cities**
   - Identify the day with the lowest temperature across all cities.
10. **Create a New Column for Wind Chill**
    - Add a new column `wind_chill` that estimates the wind chill based on the formula: $$ \text{Wind Chill} = 13.12 + 0.6215 \times \text{Temperature} - 11.37 \times (\text{Wind Speed}^{0.16}) + 0.3965 \times \text{Temperature} \times (\text{Wind Speed}^{0.16}) $$
    - (Assume `wind_speed_kph` is the wind speed in kilometers per hour.)

Dataset: Airline Flight Data

This dataset contains information about flights, including details like the airline, flight number, departure and arrival times, delays, and the distance traveled.

Sample Data:

```
flight_id,airline,flight_number,origin,destination,departure_time,arrival_time,delay_mir

1,Delta,DL123,JFK,LAX,08:00,11:00,30,3970,2023-07-01
2,United,UA456,SFO,ORD,09:30,15:00,45,2960,2023-07-01
3,Southwest,SW789,DAL,ATL,06:00,08:30,0,1150,2023-07-01
4,Delta,DL124,LAX,JFK,12:00,20:00,20,3970,2023-07-02
5,American,AA101,MIA,DEN,07:00,10:00,15,2770,2023-07-02
6,United,UA457,ORD,SFO,11:00,14:30,0,2960,2023-07-02
7,JetBlue,JB302,BOS,LAX,06:30,09:45,10,4180,2023-07-03
```

```
8,American,AA102,DEN,MIA,11:00,14:00,25,2770,2023-07-03
9,Southwest,SW790,ATL,DAL,09:00,11:00,5,1150,2023-07-03
10,Delta,DL125,JFK,SEA,13:00,17:00,0,3900,2023-07-04
```

Exercises:

1. **Find the Total Distance Traveled by Each Airline**
   - Group the data by `airline` and calculate the total distance traveled for each airline.
2. **Filter Flights with Delays Greater than 30 Minutes**
   - Filter the dataset to show only flights where the delay was greater than 30 minutes.
3. **Find the Flight with the Longest Distance**
   - Identify the flight that covered the longest distance.
4. **Calculate the Average Delay Time for Each Airline**
   - Group the data by `airline` and calculate the average delay time in minutes for each airline.
5. **Identify Flights That Were Not Delayed**
   - Filter the dataset to show only flights with `delay_minutes = 0`.
6. **Find the Top 3 Most Frequent Routes**
   - Group the data by `origin` and `destination` to find the top 3 most frequent flight routes.
7. **Calculate the Total Number of Flights per Day**
   - Group the data by `date` and calculate the total number of flights on each day.
8. **Find the Airline with the Most Flights**
   - Identify the airline that operated the most flights.
9. **Calculate the Average Flight Distance per Day**
   - Group the data by `date` and calculate the average flight distance for each day.
10. **Create a New Column for On-Time Status**
    - Add a new column called `on_time` that indicates whether a flight was on time (`True` if `delay_minutes = 0`, otherwise `False`).
