Name: Pinwen Mu

Table of Contents

Project Direction Overview

When I took the CS521, I completed my final project called Pomodoro-Workout clock. The Pomodoro Technique is a popular time management method that involves working in focused 25-minute stretches, followed by 5-minute breaks. After four pomodoros, take a longer break of 20 minutes. This method is effective in enhancing productivity and concentration. With remote work becoming more prevalent, it's easy to get distracted during breaks. When I work at home, I can't help playing my phone at the breaks, which defeats the purpose of taking a break. To address this issue, I developed a Pomodoro-Workout Clock that incorporates full body stretches and exercises to help you relax and re-energize during breaks. Of course, if you prefer to rest, you can simply choose to lie down.

That program stores all the usage records in csv files, and the data it need, like quotes, is stored in a txt file. Also, workout videos' addresses are directly written in the program. It is inconvenient to modify the data and count the records. Also, the program was designed for myself so it could not support multiple accounts. I want to use database to improve the program in the following aspects and develop the program into an app.

1. the program can support multiple accounts.
2. The users can create groups to support and encourage each other
3. Groups can participate some competition and win a reward
4. Easy to show a lot of records statistics, such as how many time you have studied or workout today, this week, this month; whether you achieve the goal you set ; how many days you have insisted on focusing, etc .
5. Easy to update some new quotes and workout videos.

Here is a brief examples of how someone would use the app. Just install the app and register or login the account . Set the goal pomodoros user wants to complete today. Click 'Start' to start a 25-minute Pomodoro clock, followed by a 5-minute break. During the break, you can choose to 'Lie Down' or 'Stretch' following the stretching video. After completing four Pomodoros, take a 20- minute break. During this longer break, you can choose to 'Lie Down' or have a 'Workout' following the workout video. If you like or dislike the videos, you can make a review. When users click 'Finish' and exit the app, it will show you a quote to encourage the user. User can see all focus and exercise records and statistics in 'Records' page. Meanwhile, the users can create a group to motivate and encourage each other. There will be some competitions for the groups, the winner group which complete more pomodoros in the certain periods will get the reward.

The database will store the basic information of each accounts, like name ,email, phone number. The accounts will be associated with the usage records, including the type of time usage(focus, workout, rest), the start and end time of each usage, the goal. It also stores the address of stretching and intense workout videos, as well as the encouraging quotes.

Use Cases and Fields

Account Signup/Installation Use Case

1. The person visits app store and installs the application.
2. The application asks them to create an account when its first run.

3. The user enters their information and the account is created in the database.
From a database perspective, this use case requires storing information about accounts (from steps #2 and #3). Steps #1 apply to the user and application but not the database directly. Significant fields for an account for this application are listed in the table below.

Field	What it Stores	Why it's Needed
account_id	The unique id of each account	To distinguish different account
account_name	Any cute or cool name users preferred	Name of the account
email	Email of the users	Maybe send some weekly report, congratulate the user on his birthday, encourage them how many days they have insisted or how many pomodoros they have completed
gender	Gender of the customers	To show them on user's home page
discription	The introduce customers want to show	To show them on user's home page, they can write anything they like
birthday	The birthday of customers	Maybe congratulate the user on his birthday, give them some coupons
group_id	The id the the group if user joined	To show the group that user joined
member_since	The date that users registered	To track how many days they have insisted

Goal Setup Use Case
After signup, the user set how many pomodoros he wants to complete today. From a database perspective, this use case requires storing information about the goal the user sets. Significant fields for an account for this application are listed in the table below.

Field	What it Stores	Why it's Needed
goal_id	The unique id of each goal settings	To distinguish different goal settings
account	The account_id	To distinguish which user sets the goal
date	The date of today	Convenient to tell whether users have achieved their goals of the date
goal	The total pomodoros that customers set for today's focus	Set a goal for every day

Start A Pomodoro Clock Period Use Case
User can click 'Start' to start a 25-minute focus period. If user complete the 25min focus, they will get one pomodoro. If they exit in the middle, they will not get the pomodoro, but how long they have

studied will still be recorded.After a focus period, user can click ‘Start ’to start a 5-minute short rest. User can choose to ‘Lie Down' or ‘Stretch ’following the stretching video. After completing 4 focus periods, there will be a 20-minute long break. During this longer break, User can choose to ‘Lie Down ’or have a ‘Workout ’following the workout video.
A pomodoro clock period is a focus period or a rest period. From a database perspective, this use case requires storing the start time and end time of each period, whether the user complete the focus period, and what does they do in the rest period. Significant fields for an account for this application are listed in the table below.

Field	What it Stores	Why it’s Needed
period_id	The unique id of each periods’ records	To distinguish different periods’ records
start_time	The time when customers start one pomodoro clock	It is convenient to count time of different types
end_time	The time when customers end one pomodoro clock	It is convenient to count time of different types
is_pomo	If customer finishes one 25-minute focus period, stores 1, or stores 0.	It is convenient to count how many pomodoros users have completed and show whether they have completed their goal
user_id	accoumt_id	To distinguish the owner of each periods’ records
focus_label	User enters what they will do in this focus period: study, work, reading, practicing piano, etc.	To record what user did in focus periods
is_workout	If customer choose ‘stretch’ or ‘workout’ during the rest period, stores 1, or stores 0	To distinguish what users do in the rest period

Fitness Video Information Use Case

When user choose ‘Stretch’ or ‘Workout’ during the rest time, the app will display correspond video for user to follow. From a database perspective, the database needs to store the information of the workout videos. A fitness video is a stretch video or a workout video. Significant fields for an account for this application are listed in the table below.

Field	What it Stores	Why it’s Needed
video_id	The unique id of each workout video information	To distinguish different fitness video information
video_name	the name of the vedio	Different workout for different rest time
video_address	The internet address of the workout video	Users can workout during the rest time following the video

intensity	Stores the intensity of the workout videos: 1 is low, 2 is middle, 3 is high	Users can choose the intensity they want when they choose 'workout'
body_part	Store which body part the stretch videos focus on: upper body, lower body, full body	Users can choose which part they want to relax when they choose 'stretch'

Video Reviews Use Case

After the workout, user can make a review of the video whether they like or dislike it . Significant fields for an account for this application are listed in the table below.

Field	What it Stores	Why it's Needed
review_id	A unique identifier for the review	A unique identifier for the review
video_id	The id of the video	This specifies which video the review pertains to.
account_id	The id of the account	This identifies which user submitted the review.
like_dislike	This can be an enumerated type like "Like"or"Dislike"	Used to capture a user's preference for the video.
comment	A textual review from the user.	This can be optional, as some users might just want to provide a 'like or dislike' without a written comment.
review_date	The date and time the review was submitted.	The date and time the review was submitted.

Group Use Case

Field	What it Stores	Why it's Needed
group_id	A unique identifier for the group.	To distinguish different groups
group_name	The name of the group.	The name of the group.
creation_date	The date when the group was created.	The date when the group was created.
total_pomodoros	The cumulative number of pomodoros completed by the group	To show the achievement of the group, maybe some prize for getting certain pomodoros
group_description	A textual description of the group	Provide more information about the group's purpose, goals, or any other relevant details.

User can create or join a group to encourage and company with each other. Significant fields for an account for this application are listed in the table below.

Group Competition Use Case

There will be one competition every month, the winner will get the prize. Significant fields for the competition information are listed in the table below.

Field	What it Stores	Why it's Needed
competition_id	A unique identifier for the competition	A unique identifier for the competition
competition_name	The name of the competition	To tell what the competition is.
competition_description	Detailed information about the competition.	This can describe the nature of the competition, any rules, prize, or other specifics.
start_date	The date and time when the competition starts.	The date and time when the competition starts.
end_date	The date and time when the competition starts.	The date and time when the competition starts.
prize_id	The prize_id	The foreign key refer to the prize

Prize Use Case

There will be some different prizes for the competitions. Every competition will choose one prize for the winner.

Field	What it Stores	Why it's Needed
prize_id	A unique identifier for the prize	A unique identifier for the prize
prize_name	The name of the prize	To tell what the prize is.
prize_description	Detailed information about the prize.	This can describe the nature of the reward, any conditions, when to release, or other specifics.
prize_value	The number of prize's value	If the prize has a tangible value, like cash, points, this attribute can capture that value.

Matches Use Case

If Groups participate the competition, one group will have a match with another , the winner will be the group who completed more pomodoros in the competition period. Significant fields for matches are listed in the table below.

Field	What it Stores	Why it's Needed
competition_id	A unique identifier for the competition	A unique identifier for the competition
group1_id	The id of the first group	the first group participating in the competition
group2_id	The id of the second group	the second group participating in the competition

group1_pomodoros	The number of pomodoros completed by the first group during the competition	To show the real-time achievements of group1
group2_pomodoros	The number of pomodoros completed by the second group during the competition.	To show the real-time achievements of group2
winner_group	The group ID of the winning group	To show which group win the competition

Prize Value Change Use Case

For every prize, their value may be change over time. These changes may be involved in the cost of this app. So we need to record the value change of the prize. Significant fields for matches are listed in the table below.

Field	What it Stores	Why it's Needed
change_id	A unique identifier for the change record	A unique identifier for the change record
prize_id	The id of the prize	To indicate which prize has the value change
old_value	The old value of the prize	To indicate the old value of the prize before change
new_value	The new value of the prize	To indicate the new value of the prize after change
date	Current date	The date to commit the value change

Structural Database Rules

1. According to the Account Signup/Installation Use Case, there is only one entity: account. I do not see any other entity or relationship. A pomodoro clock period is a focus period or a rest period.

2.According to the Goal Setup Use Case, there are two entities : account and goal. A user will set a goal for the day when they use the pomodoro-workout app , but a goal is must be set by a user.
The rue here is: Each account may set many goals; each goal must be set by an account.

3.According to the Start A Pomodoro Clock Period Use Case, there are two entities : account and the pomodoro_clock_period. A user may use many pomodoro_clock_periods over time , but each pomodoro_clock_period is used by one user.
The rule here is: Each account may use many pomodoro_clock_period; each pomodoro_clock_period must be used by an account.

4.According to the Fitness Video Information Use Case, there is only one entity: fitness video. I do not see any other entity or relationship. A fitness video is a stretch video or a workout video.

5.According to the Video Reviews Use Case, there are three entities : workout video, review, account. A user may make reviews on many workout videos.
The rule here is :Each account may write many reviews; Each review must be written by an account. Each workout video may have many reviews; Each review must be associated with a workout video.

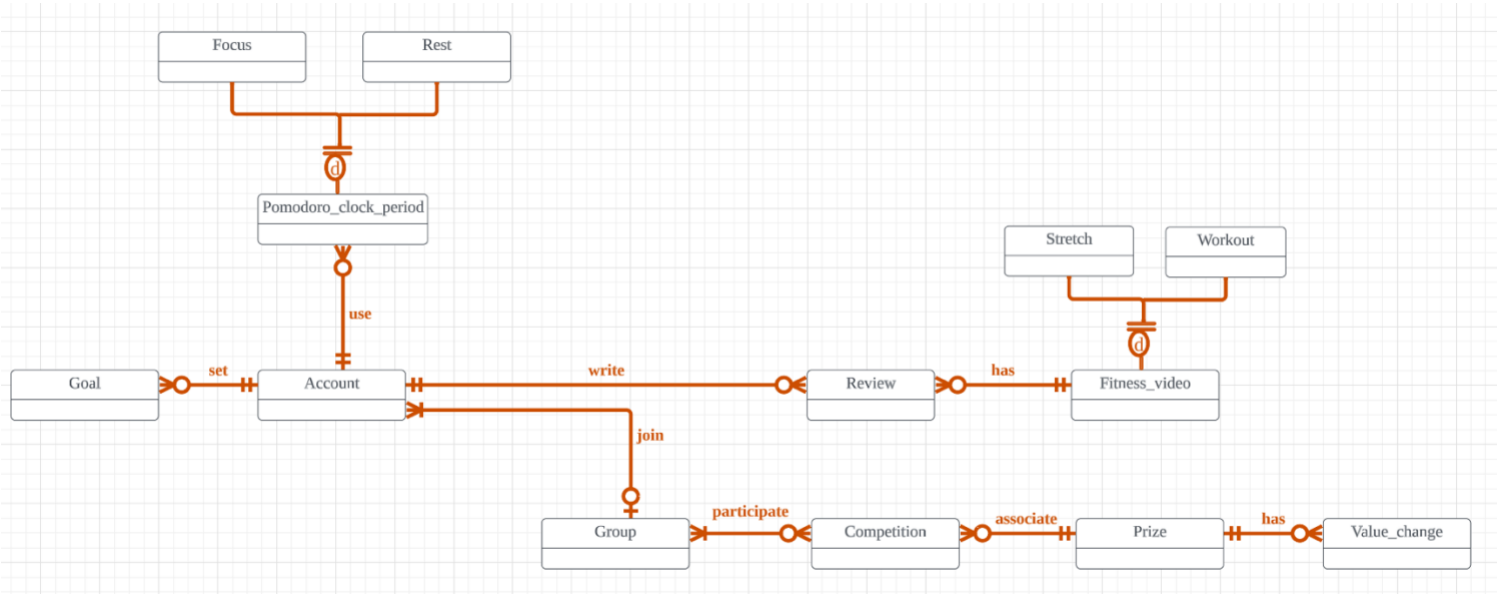
6.According to the Group Use Case, there are two entities: account and group. A user can join only one group, a group must be joined by 3 accounts.

The rule here is: Each account may join one group; each group must be joined by many accounts.
7.According to the Group Competition Use Case, there are two entities: competition and prize. Every competition has a kind of prize for the winner and the prize may be used in different competitions.
The rule here is: Each competition must have one prize; Each prize may be associated with many competitions.
8.According to the Prize Use Case, there is only one entity: prize. I don't see any other entity or relationship .
9.According to the Matches Use Case, there are two entities: group and competition.A group may participate many competitions and a competition must be participated by many groups.
The rule here is :Each group may participate many competition; each competition must be participated by many groups.
10. According to the Prize Value Change Use Case, there are two entities: Prize and Value_change. A prize may have different value changes over time; each value change must be related to one prize.

From the use cases I have, there are I have these 10 associative structural database rules.

(1) Each account may set many goals; each goal must be set by an account.
(2) Each account may use many pomodoro_clock_periods; each pomodoro_clock_period must be used by an account.
(3) Each account may write many reviews; Each review must be written by an account.
(4) Each workout video may have many reviews; Each review must be associated with a workout video.
(5) Each account may join one group; each group must be joined by many accounts.
(6) Each competition must be associated with one prize; Each prize may be associated with many competitions
(7) Each group may participate many competition; each competition must be participated by many groups.
(8) A pomodoro clock period is a focus period or a rest period.
(9) A fitness video is a stretch video or a workout video.
(10) A prize may have different value changes over time; each value change must be related to one prize.

Conceptual Entity-Relationship Diagram



Full DBMS Physical ERD

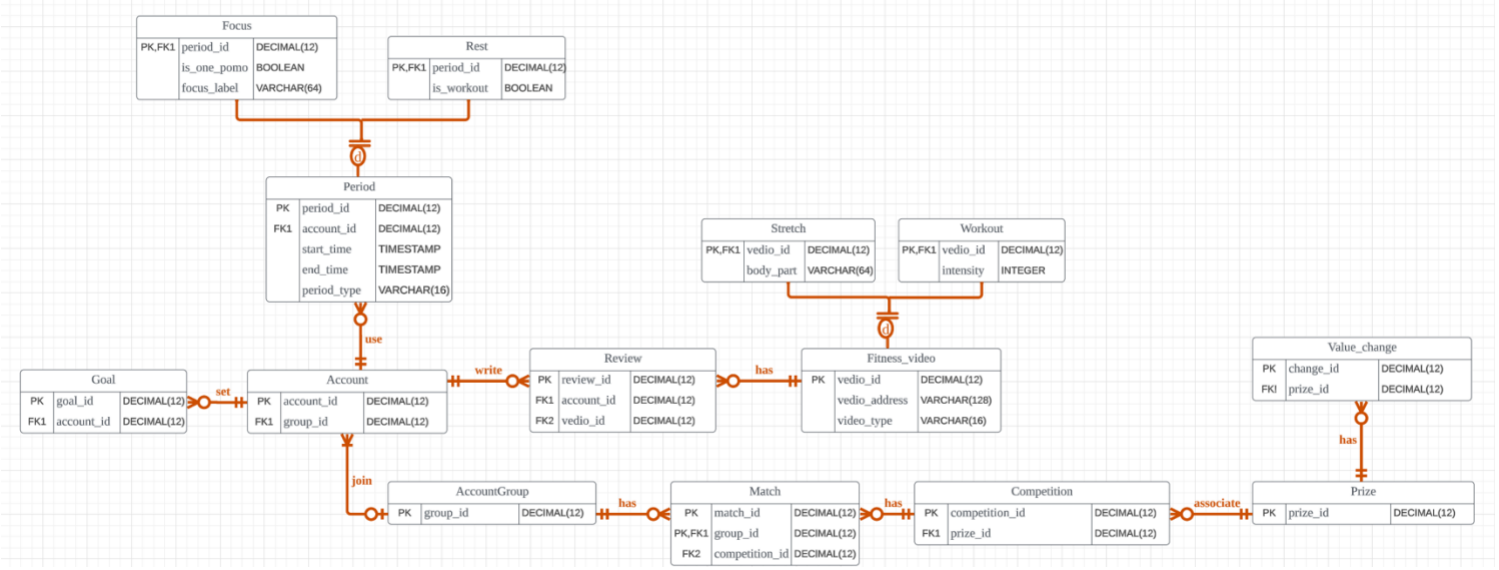
In my conceptual ERD, Account/Goal, Account/Pomodoro_clock_period, Account/Review, Fitness_video/Review, Prize/Competition, Group/Account are 1:M relationships, and Group/Competition is a M:N relationship.For the 1:M relationships, I retained the entities from the conceptual ERD.

Since Group/Competition is an M:N relationship, it was necessary for me to create a bridging entity to support the relationship. I named that entity 'Match' to give it a useful name in the database. The bridging entity has foreign keys to both Group and Competition, resulting in two 1:M relationships between Match and Group and Competition.

I also have two specialization-generalization relationships in my conceptual ERD, one for the Pomodoro_clock_period entity and one for the Fitness_video entity. The additional entities under Pomodoro_clock_period are Focus and Rest, each of which have a primary and foreign key of period_id which reference the primary key of Pomodoro_clock_period. The Focus subtype has two attributes, is_pomo stores 1 if the user complete the 25-min focus period and get one pomodoro, if user exit in the middle then is_pomo will store 0 indicating user doesn't get one pomodoro, and the focus_label attribute will stores the user's input about what they will do during this following focus period, such as reading, piano, study, etc. The Rest subtype has one attribute, is_workout stores 1 if the user choose 'Stretch' or 'Workout' during the rest period and stores 0 if user choose 'Lie Down'.

The additional entities under Fitness_video are Stretch and Workout, which have primary and foreign keys of vedio_id which reference the primary key of Fitness_vedio. The Stretch has one attribute: body_part. It will store which body part the stretch videos focus on, upper body, lower body or full body. The attribute of Workout subtype is intensity, which Stores the intensity of the workout videos, 1 is low, 2 is middle, 3 is high.

Here is my initial DBMS physical ERD with these relationships mapped into them.



Pomodoro-Workout Clock Attributes

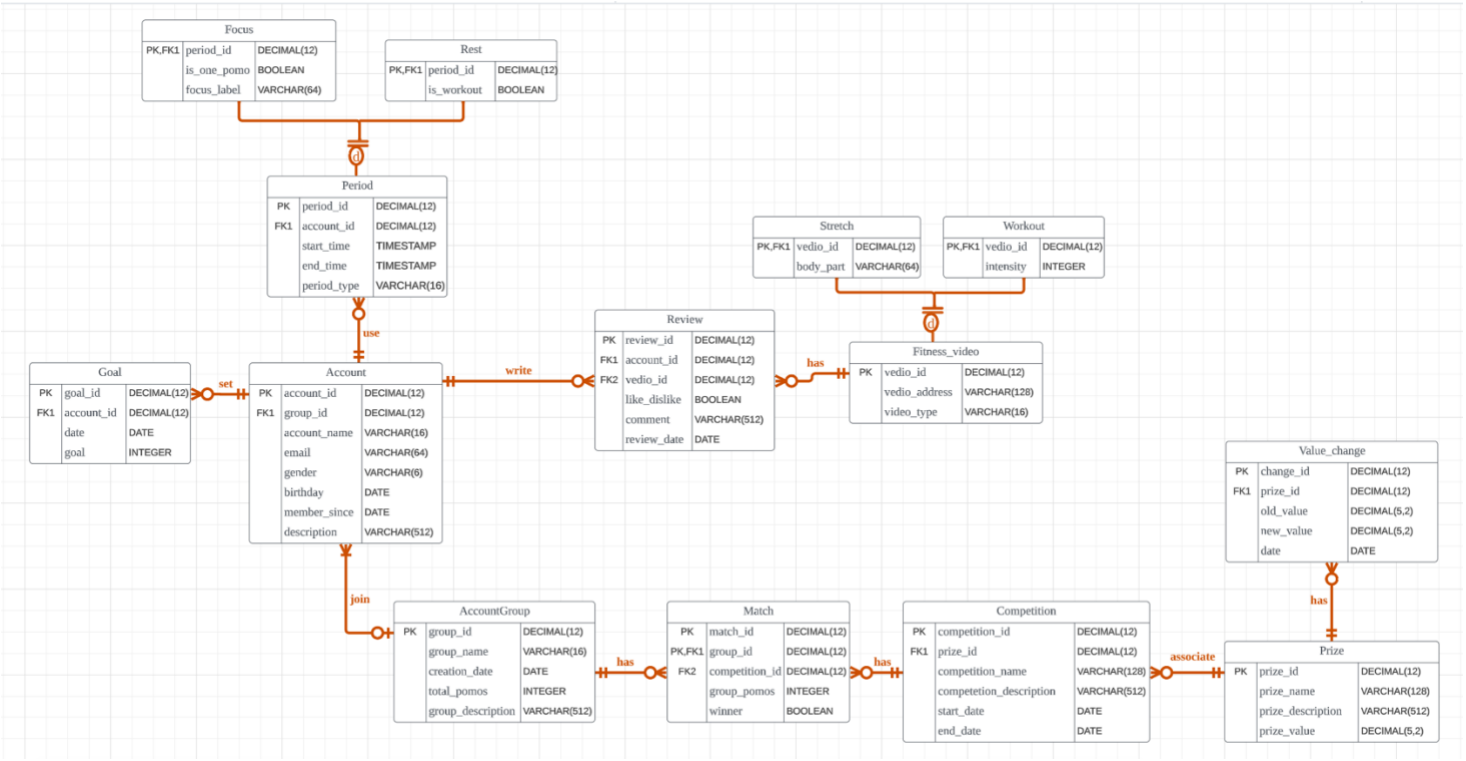
To make it easier to create tables in PostgreSQL, I changed the table name "group" to "AccountGroup" and the table name "Pomodoro" to "period". Also, I add ‘period_type’ in table Period and ‘video_type’ in table Fitness_video as discriminators. For the Match table, I set the group_id and match_id as a synthetic primary key. I am now going through the process of adding attributes and their datatypes table-by-table. My choices and reasoning are in the table below

Table	Attribute	Datatype	Reasoning	Example Data
Account	account_name	VARCHAR(16)	This is just the name of the account, so I don’t think it need pretty long of the name,16 is enough.	MAKABAKA
Account	email	VARCHAR(64)	This is the email of the users, I think 64 characters will be enough	makabaka@gmail.com
Account	gender	VARCHAR(6)	Gender of the users, male or female, so I give it 6 characters	female
Account	description	VARCHAR(512)	The introduce users want to show on their homepage, I think maybe users have a lot to express, so I give it 512 characters	Draco Dormiens Nunquam Titillandus
Account	birthday	DATE	The birthday of users, so the type is DATE	1993/10/27
Account	member_since	DATE	The date that users registered, so the type is DATE	2022/10/1
Goal	date	DATE	The date of the current day, so the type is DATE	2023/3/1
Goal	goal	INTEGER	The total pomodoros that users set for today’s focus, so the type is integer.	4
Period	start_time	TIMESTAMP	The time when users start one period, so the type is TIMESTAMP.It is convenient to count the time period based on minutes.	2023/7/15 00:00
Period	end_time	TIMESTAMP	The time when users end one period, so the type is TIMESTAMP.It is convenient to count the time period based on minutes.	2023/7/15 00:25
Period	period_type	VARCHAR(16)	It’s a discriminator of Period’s subtype: focus or rest, so I give it 16 characters. It is enough for the subtype name even I need to change the subtype name.	focus
Focus	is_one_pomo	BOOLEAN	If customer finishes one 25-minute focus period, stores 1, or stores 0, so the datatype is BOOLEAN	1
Focus	focus_label	VARCHAR(64)	User enters what they will do in this focus period: study, work, reading, practicing piano, etc. I think 64 characters is enough to label.	reading

Rest	is_workout	BOOLEAN	If customer choose 'stretch' or 'workout' during the rest period, stores 1, or stores 0,so the date type is boolean	0
Fitness_vedio	video_type	VARCHAR(16)	It's a discriminator of Fitness_vedio's subtype: focus or rest, so I give it 16 characters. It is enough for the subtype name even I need to change the subtype name.	workout
Fitness_vedio	video_address	VARCHAR(128)	The internet address of the workout video. I think 128 characters are enough to store.	https://www.youtube.com/watch?v=UltWltVZZmE&t=1s
Workout	intensity	INTEGER	Stores the intensity of the workout videos: 1 is low, 2 is middle, 3 is high, so the datatype is integer.	2
Stretch	body_part	VARCHAR(64)	Store which body part the stretch videos focus on: upper body, lower body, full body, so I give 64 characters to store.	upper body
Review	like_dislike	BOOLEAN	If the users like the video, stores 1, or stores 0, so I set the datatype boolean.	0
Review	comment	VARCHAR(512)	A textual review from the user on the fitness videos, I think 512 characters will be enough.	The intensity is too low
Review	review_date	DATE	The date the review was submitted, so the datatype is DATE	2023/3/1
AccountGroup	group_name	VARCHAR(16)	The name of the group. I think 16 characters will be enough.	KiaOra
AccountGroup	creation_date	DATE	The date when the group was created, so the datatype is DATE	2023/3/1
AccountGroup	total_pomodoros	INTEGER	The cumulative number of pomodoros completed by the group, so the datatype is integer.	54
AccountGroup	group_description	VARCHAR(512)	Provide more information about the group's purpose, goals, or any other relevant details. I think 512 characters are enough.	Together We Achieve More
Competition	competition_name	VARCHAR(128)	The name of the competition. Some competition's name may be long, so I set 128 characters.	Summer Ice Cream Cup Competition
Competition	competition_description	VARCHAR(512)	Detailed information about the competition. I set 512 characters.	The team that collects the most pomodoros from the start date to the end date will be crowned winners. Winners will receive an ice cream voucher valued at 20 dollar.
Competition	start_date	DATE	The date when the competition starts, so the datatype is DATE	2023/9/25
Competition	end_date	DATE	The date when the competition ends, so the datatype is DATE	2023/10/1
Prize	prize_name	VARCHAR(128)	The name of the prize, I think 128 characters are enough for some fancy name.	Häagen-Dazs ice cream voucher
Prize	prize_description	VARCHAR(512)	Detailed information about the prize. I set 512 characters.	Can be used at any Häagen-Dazs store nationwide. Valid for one year.
Prize	prize_value	DECIMAL(5,2)	The number of prize's value. Most competition is based on weekly, so the prize won't be too expensive. I think 5 digits are enough and 2 decimals to be accurate.	20.00
Match	group_pomos	INTEGER	The number of pomodoros completed by the group during the competition, so the datatype is integer.	200
Match	winner	BOOLEAN	This is to show whether this group is the winner, so the datatype is boolean, 1 for winner, 0 for not.	1

Value_change	old_value	DEIMAL(5,2)	The old value of the prize, so the data type is the same as the prize_value	20
Value_change	new_value	DEIMAL(5,2)	The new value of the prize, so the data type is the same as the prize_value	25
Value_change	date	DATE	The date of current date	2023-10-15

Here is my ERD with the attributes included.



Normalization:

There might be data redundancy in table AccountGroup if total_pomos' data is changed and inserted. But this attribute is to store the cumulative number of pomodoros completed by the group to show their great effort, so I don't need to know the history data of this attribute. It will be real-time update with a trigger so there won't have data redundancy. And same as the attribute 'group_pomos' in table Match.

Stored Procedure Execution and Explanations

In the Start A Pomodoro Clock Period Use Case, User can click 'Start' to start a 25-minute focus period. If user complete the 25min focus, they will get one pomodoro. If they exit in the middle, they will not get the pomodoro, but how long they have studied will still be recorded.After a focus period, user can click 'Start 'to start a 5-minute short rest. User can choose to 'Lie Down' or 'Stretch 'following the stretching video. After completing 4 focus periods, there will be a 20-minute long break. During this longer break, User can choose to 'Lie Down 'or have a 'Workout 'following the workout video.A pomodoro clock period is a focus period or a rest period.

For this use case, I will implement a transaction that inserts period information into the Focus table and the Rest table. Here is a screenshot of my stored procedure definition.

Query Query History

```
168 -- Insert new Focus period
169 CREATE OR REPLACE PROCEDURE insert_focus_period(v_account_name VARCHAR(16),v_start_time TIMESTAMP,
170 v_end_time TIMESTAMP,v_focus_label VARCHAR(64))
171 LANGUAGE plpgsql
172 AS $$
173 DECLARE
174     v_is_one_pomo BOOLEAN;
175     v_duration_seconds INTEGER;
176     v_account_id DECIMAL(12);
177 BEGIN
178     -- Parameter Validation
179     IF v_account_name IS NULL OR v_start_time IS NULL OR v_end_time IS NULL OR v_focus_label IS NULL THEN
180         RAISE EXCEPTION 'All parameters must be provided';
181     END IF;
182     -- Calculate the duration in seconds
183     v_duration_seconds := EXTRACT(EPOCH FROM (v_end_time - v_start_time));
184     -- Check if the duration is exactly 25 minutes (1500 seconds)
185     IF v_duration_seconds = 1500 THEN
186         v_is_one_pomo := TRUE;
187     ELSE
188         v_is_one_pomo := FALSE;
189     END IF;
190     -- Get the account_id using account_name
191     SELECT INTO v_account_id account_id FROM Account WHERE account_name = v_account_name;
192     IF v_account_id IS NULL THEN
193         RAISE EXCEPTION 'Account with the given name does not exist';
194     END IF;
195     -- Insert the focus period record
196     INSERT INTO Period(period_id, account_id, start_time, end_time, period_type)
197     VALUES (NEXTVAL('period_seq'), v_account_id, v_start_time, v_end_time, 'focus');
198     -- Insert the focus record
199     INSERT INTO Focus(period_id, is_one_pomo, focus_label)
200     VALUES (CURRVAL('period_seq'), v_is_one_pomo, v_focus_label);
201 END;
202 $$;
203
```

There is an attribute in Focus, is_one_pomo, this attribute stores true if this focus period last for 25minutes, so in the insert_focus_period procedure, I write an if-else clause to determine whether "is_one_pomo" is true by comparing whether the period is less than 25 min.

```sql
-- Insert new Rest period
CREATE OR REPLACE PROCEDURE insert_rest_period(
    v_account_name VARCHAR(16),
    v_start_time TIMESTAMP,
    v_end_time TIMESTAMP,
    v_is_workout BOOLEAN
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_account_id DECIMAL(12);
BEGIN
    -- Parameter Validation
    IF v_account_name IS NULL OR v_start_time IS NULL OR v_end_time IS NULL THEN
        RAISE EXCEPTION 'All parameters (account_name, start_time, and end_time) must be provided';
    END IF;
    -- Get the account_id using account_name
    SELECT INTO v_account_id account_id FROM Account WHERE account_name = v_account_name;
    IF v_account_id IS NULL THEN
        RAISE EXCEPTION 'Account with the given name does not exist';
    END IF;
    -- Insert the rest period record
    INSERT INTO Period(period_id, account_id, start_time, end_time, period_type)
    VALUES (NEXTVAL('period_seq'), v_account_id, v_start_time, v_end_time, 'rest');
    -- Insert the rest record
    INSERT INTO Rest(period_id, is_workout)
    VALUES (CURRVAL('period_seq'), v_is_workout);

END
$$;
```

I give them parameters that correspond to the Period and Focus/Rest tables. Since user name is easier for looking up, so I change the parameter account_id to account_name and use select clause in the procedure to get the correspond account_id. Since both focus and rest are the subtype of the Period, so I insert the period record first, then insert into the focus and rest records.Since this procedure is always for a Focus/Rest period, I hardcode the period_type in Period with focus/rest.
Here is a screenshot of my stored procedure execution.

```sql
-- Insert Focus and Rest period records for Charlie
START TRANSACTION;
DO
$$BEGIN
   CALL insert_focus_period('Charlie', '2023-10-04 08:00:00', '2023-10-04 08:25:00', 'Study');
END$$;
DO
$$BEGIN
   CALL insert_rest_period('Charlie', '2023-10-04 08:25:00', '2023-10-04 08:30:00', true);
END$$;
-- Commit the transaction
COMMIT;
```

Data Output | Notifications | Messages

```
COMMIT

Query returned successfully in 232 msec.
```

I insert focus and rest periods for each 15 group members in the database.

In the Fitness Video Information Use Case, When user choose ‘Stretch’ or ‘Workout’ during the rest time, the app will display correspond video for user to follow. From a database perspective, the

database needs to store the information of the workout videos. A fitness video is a stretch video or a workout video.
For this use case, I will implement a transaction that inserts stretch and workout video information into the Stretch table and the Workout table. Here is a screenshot of my stored procedure definition.

```
214  -- Insert new Stretch video
215  CREATE OR REPLACE PROCEDURE insert_stretch_video(
216      v_video_address VARCHAR,
217      v_body_part VARCHAR
218  )
219  LANGUAGE plpgsql AS $$
220  DECLARE
221      new_video_id DECIMAL;
222  BEGIN
223      -- Get new vedio_id from the sequence
224      new_video_id := NEXTVAL('fitness_video_seq');
225
226      INSERT INTO Fitness_video(vedio_id, vedio_address, vedio_type)
227      VALUES (new_video_id, v_video_address, 'Stretch');
228
229      INSERT INTO Stretch(vedio_id, body_part)
230      VALUES (new_video_id, v_body_part);
231  END;
232  $$;
233
```

```
234  -- Insert new Workout video
235  CREATE OR REPLACE PROCEDURE insert_workout_video(
236      v_video_address VARCHAR,
237      v_intensity INTEGER
238  )
239  LANGUAGE plpgsql AS $$
240  DECLARE
241      new_video_id DECIMAL;
242  BEGIN
243      -- Get new vedio_id from the sequence
244      new_video_id := NEXTVAL('fitness_video_seq');
245
246      INSERT INTO Fitness_video(vedio_id, vedio_address, vedio_type)
247      VALUES (new_video_id, v_video_address, 'Workout');
248
249      INSERT INTO Workout(vedio_id, intensity)
250      VALUES (new_video_id, v_intensity);
251  END;
252  $$;
253
```

I name the stored procedure “insert_stretch_video” and ”insert_workout_video”, and give it parameters that correspond to the Fitness and Stretch/Workout tables. Since this procedure is always for a stretch/workout video, I hardcode the video_type in Fitness_vedio table with stretch/workout.

Here is a screenshot of my stored procedure execution.

```
750
751  START TRANSACTION;
752  DO
753  $$BEGIN
754     CALL insert_stretch_video('https://www.youtube.com/watch?v=gfzC9XMEypg', 'full body');
755  END$$;
756  COMMIT TRANSACTION;
757
```

Data Output | Notifications | Messages

```
COMMIT

Query returned successfully in 389 msec.
```

```
783
784  START TRANSACTION;
785  DO
786  $$BEGIN
787     CALL insert_workout_video('https://www.youtube.com/watch?v=6Bjtq2A_jwg', 3);
788  END$$;
789  COMMIT TRANSACTION;
790
791  START TRANSACTION;
792  DO
793  $$BEGIN
794     CALL insert_workout_video('https://www.youtube.com/watch?v=J4wm6qiv5pI', 3);
795  END$$;
796  COMMIT TRANSACTION;
```

Data Output | Notifications | Messages

```
COMMIT

Query returned successfully in 345 msec.
```

I insert 5 records for both stretch table and workout table in the database.

I also create a trigger on focus table to total_pomos in table AccountGroup. The total_pomos is the sum of the group members' pomos. One Pomodoro is competed by a group member, the is_one_pomo of the focus period will record true, and the total_pomos should add one.

```sql
-- Create a trigger on table focus to update the total_pomos in table AccountGroup
CREATE OR REPLACE FUNCTION update_total_pomos()
RETURNS TRIGGER AS $$
DECLARE
    v_group_id DECIMAL(12);
    v_total_pomos INTEGER;
BEGIN
    -- Get the group_id associated with the inserted Focus row
    SELECT group_id INTO v_group_id FROM Account
    WHERE account_id = (SELECT account_id from Period WHERE period_id=New.period_id);
    -- Calculate the total count of Focus rows with is_one_pomo equal to 1
    SELECT COUNT(*) INTO v_total_pomos
    FROM Focus
    JOIN Period on Period.period_id=Focus.period_id
    JOIN Account on Period.account_id=Account.account_id
    GROUP BY Account.group_id,Focus.is_one_pomo
    HAVING group_id = v_group_id AND is_one_pomo = TRUE;
    -- Update the total_pomos field in the relevant group of the AccountGroup table
    UPDATE AccountGroup SET total_pomos = v_total_pomos WHERE group_id = v_group_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
-- Create a trigger to fire when inserting new rows into the Focus table
CREATE TRIGGER focus_insert_trigger
AFTER INSERT ON Focus
FOR EACH ROW
EXECUTE FUNCTION update_total_pomos();
```

Befor the focus periods are inserted, the total_pomos of each group is 0.

```sql
DELETE FROM Focus;
SELECT * FROM AccountGroup;
```

Data Output | Notifications | Messages

	group_id [PK] numeric (12)	group_name character varying (16)	creation_date date	total_pomos integer	group_description character varying (512)
1	1	Study Group 1	2023-09-30	0	Together, we achieve more. Let's conquer our academic challenges!
2	2	Study Group 2	2023-09-29	0	Persistence guarantees that results are inevitable. Keep going!
3	3	Study Group 3	2023-09-28	0	Every study session is a step closer to success. Stay focused!
4	4	Study Group 4	2023-09-27	0	Knowledge is power, and we are its seekers. Let's empower ourselves!
5	5	Study Group 5	2023-09-26	0	Dedication and hard work are the keys to success. Let's unlock our potential togeth...

After inserting focus periods into the table Focus, the total_pomos updates by the trigger.

```
688 DO
689 $$BEGIN
690     CALL insert_rest_period('Olivia', '2023-10-02 08:55:00', '2023-10-02 09:00:00', true);
691 END$$;
692 DO
693 $$BEGIN
694     CALL insert_focus_period('Olivia', '2023-10-02 09:00:00', '2023-10-02 09:25:00', 'Study');
695 END$$;
696 DO
697 $$BEGIN
698     CALL insert_rest_period('Olivia', '2023-10-02 09:25:00', '2023-10-02 09:30:00', false);
699 END$$;
700 COMMIT;
701
702 SELECT * FROM AccountGroup;
703
```

Data Output | Notifications | Messages

	group_id [PK] numeric (12)	group_name character varying (16)	creation_date date	total_pomos integer	group_description character varying (512)
1	1	Study Group 1	2023-09-30	8	Together, we achieve more. Let's conquer our academic challenges!
2	2	Study Group 2	2023-09-29	5	Persistence guarantees that results are inevitable. Keep going!
3	3	Study Group 3	2023-09-28	4	Every study session is a step closer to success. Stay focused!
4	4	Study Group 4	2023-09-27	3	Knowledge is power, and we are its seekers. Let's empower ourselves!
5	5	Study Group 5	2023-09-26	5	Dedication and hard work are the keys to success. Let's unlock our potential togeth...

Question Identification and Explanations

Question 1: What is the winner information for each match?
Every time there is a competition, I believe many groups will participate. Two of them will have a match, the winner gets the prize. In the interest of fairness and transparency, the list of winners and competition information is typically announced after the competition concludes.

Question 2: How are the reviews for different types of fitness videos?
This app is designed to combine focus training and do some workout in the rest period to maximum the utilization of time, so the reviews on the fitness videos is important. I can change or recommend the videos due to the users reviews.

Question 3: Has the user achieved their goals set for each time?
The user will set a goal before everyday use, they must be curious about whether they complete the goal every time. It is necessary to have a query to show the competition statues of the goal.

Query Executions and Explanations

Replace this with queries answering the questions, along with screenshots and explanations.
For question 1, the query answering the question is showed as below.

```
861
862  --Q:What is the winner information for each match
863  SELECT g.group_name AS match_winner, c.competition_name,
864  c.start_date, c.end_date, m.group_pomos AS scores, p.prize_name
865  From AccountGroup AS g
866  JOIN Match AS m ON g.group_id=m.group_id
867  JOIN Competition AS c ON m.competition_id=c.competition_id
868  JOIN Prize AS p ON c.prize_id=p.prize_id
869  WHERE m.winner=true;
870
```

Data Output | Notifications | Messages

	match_winner character varying (16)	competition_name character varying (128)	start_date date	end_date date	scores integer	prize_name character varying (128)
1	Study Group 1	Competition 1	2023-10-02	2023-10-05	8	Coffee Gift Card
2	Study Group 3	Competition 1	2023-10-02	2023-10-05	3	Coffee Gift Card
3	Study Group 5	Competition 2	2023-10-06	2023-10-08	2	Bookstore Voucher

This query shows the winner of each competition in every competition, along with the prize information and the winners' scores.

For question 2, the query answering the question is showed as below.

```
870
871  --Q:How are the reviews for different types of fitness videos?
872  SELECT a.account_name,r.like_dislike,r.comment,
873  f.vedio_address,s.body_part AS stretch_type,w.intensity AS workout_intensity
874  FROM Account AS a
875  JOIN Review AS r ON a.account_id=r.account_id
876  JOIN Fitness_video AS f ON f.vedio_id=r.vedio_id
877  LEFT JOIN Stretch AS s ON f.vedio_id=s.vedio_id
878  left JOIN Workout AS w ON f.vedio_id=w.vedio_id;
```

Data Output | Notifications | Messages

	account_name character varying (16)	like_dislike boolean	comment character varying (512)	vedio_address character varying (128)	stretch_type character varying (64)	workout_intensity integer
1	Bob	false	I didn't like this video	https://www.youtube.com/watch?v=hLVh5IBsCxk	[null]	3
2	Frank	true	I liked this video	https://www.youtube.com/watch?v=sAf67xFS-qE	full body	[null]
3	Alice	true	This video is great!	https://www.youtube.com/watch?v=UYfHWsuxe50	upper body	[null]
4	Eva	false	I disliked this video	https://www.youtube.com/watch?v=Ef6LwAaB3_E	full body	[null]
5	Charlie	true	I liked this video	https://www.youtube.com/watch?v=D_x1fPDZm_A	lower body	[null]

This query shows all the review information on videos, along with the specific types of stretch vedio and workout video. From this table, we can clearly tell all the reviews information for the videos.

For question 3, the query answering the question is showed as below.

```sql
--Q:Has the user achieved their goals set for each time?
CREATE OR REPLACE VIEW GoalAchievement AS
WITH PomosCount AS (
    SELECT
        a.account_name,
        g.goal,
        g.date,
        COUNT(*) AS today_pomos
    FROM Focus AS f
    LEFT JOIN Period AS p ON f.period_id = p.period_id
    JOIN Account AS a ON p.account_id = a.account_id
    JOIN Goal AS g ON g.account_id = a.account_id
    WHERE f.is_one_pomo = true AND DATE(p.end_time) = g.date
    GROUP BY a.account_name, g.goal, g.date)

SELECT
    pc.account_name,
    pc.goal,
    pc.date,
    pc.today_pomos,
    CASE
        WHEN pc.today_pomos >= pc.goal THEN 'Yes'
        ELSE 'No'
    END AS goal_achieved
FROM PomosCount AS pc;
```

Data Output | Notifications | Messages

```
CREATE VIEW

Query returned successfully in 250 msec.
```

I create a view called GoalAchievement. First, I create a view of PomosCount. It retrieve information from table Focus, Period ,Goal and Account to count the pomodoros the user completed in that day(consistency with the date of the goal).Then, in the view of GoalAchievement, I compare the today_pomo with the user's goal on that day to tell if the user compete the goal.

Query | Query History

```
1030
1031  SELECT * FROM GoalAchievement
1032  ORDER BY account_name,date;
```

Data Output | Notifications | Messages

	account_name character varying (16)	goal integer	date date	today_pomos bigint	goal_achieved text
1	Alice	2	2023-10-02	3	Yes
2	Bob	3	2023-10-03	4	Yes
3	Charlie	1	2023-10-04	1	Yes
4	David	2	2023-10-05	3	Yes
5	Eva	3	2023-10-06	1	No
6	Frank	4	2023-10-07	1	No
7	Grace	2	2023-10-08	1	No
8	Hugo	3	2023-10-02	2	No
9	Ivy	1	2023-10-03	1	Yes
10	Jack	2	2023-10-04	1	No
11	Kara	3	2023-10-05	1	No
12	Leo	4	2023-10-06	1	No
13	Mia	2	2023-10-07	1	No
14	Nate	3	2023-10-08	1	No
15	Olivia	1	2023-10-02	3	Yes
16	Paul	1	2023-10-02	3	Yes
17	Paul	2	2023-10-03	1	No
18	Paul	3	2023-10-04	1	No
19	Paul	1	2023-10-05	2	Yes
20	Paul	2	2023-10-06	1	No
21	Paul	3	2023-10-07	1	No
22	Paul	1	2023-10-08	1	Yes

Also, the GoalAchievement can be looked up by user names or the achievement.

```
907  SELECT * FROM GoalAchievement
908  WHERE account_name='Bob';
```

Data Output | Notifications | Messages

	account_name character varying (16)	goal integer	date date	today_pomos bigint	goal_achieved text
1	Bob	3	2023-10-03	4	Yes

```
1034  SELECT * FROM GoalAchievement
1035  WHERE goal_achieved='Yes';
```

Data Output | Notifications | Messages

	account_name character varying (16)	goal integer	date date	today_pomos bigint	goal_achieved text
1	Alice	2	2023-10-02	3	Yes
2	Bob	3	2023-10-03	4	Yes
3	Charlie	1	2023-10-04	1	Yes
4	David	2	2023-10-05	3	Yes
5	Ivy	1	2023-10-03	1	Yes
6	Olivia	1	2023-10-02	3	Yes
7	Paul	1	2023-10-02	3	Yes
8	Paul	1	2023-10-05	2	Yes
9	Paul	1	2023-10-08	1	Yes

Index Identification and Creations

Replace this with indexes identifications useful to your database, explanations as to why they help, and screenshots of their creations.
As far as primary keys which are already indexed, here is the list.
Focus.period_id
Rest.period_id
Period.period_id
Goal.goal_id
Account.account_id

Review.review_id
Fitness_video.vedio_id
Stretch.video_id
Workout.video_id
AccountGroup.group_id
Match.match_id
Match.group_id
Competition.competition_id

As far as foreign keys, I know all of them need an index. Below is a table identifying each foreign key column, whether or not the index should be unique or not, and why.

Column	Unique?	Description
Period.account_id	Not unique	there can be many period records of the same user
Goal.account_id	Not unique	there can be many goal records of the same user
Account.group_id	Not unique	Three different account may be in one group, with the same group id
Review.account_id	Not unique	there can be many reviews fron the same user
Review.video_id	Not unique	there can be many reviews no the same videos
Match.competition_id	Not unique	there can be many matches in the same one competition
Competition.prize_id	Not unique	there can be many competitions have the same prize

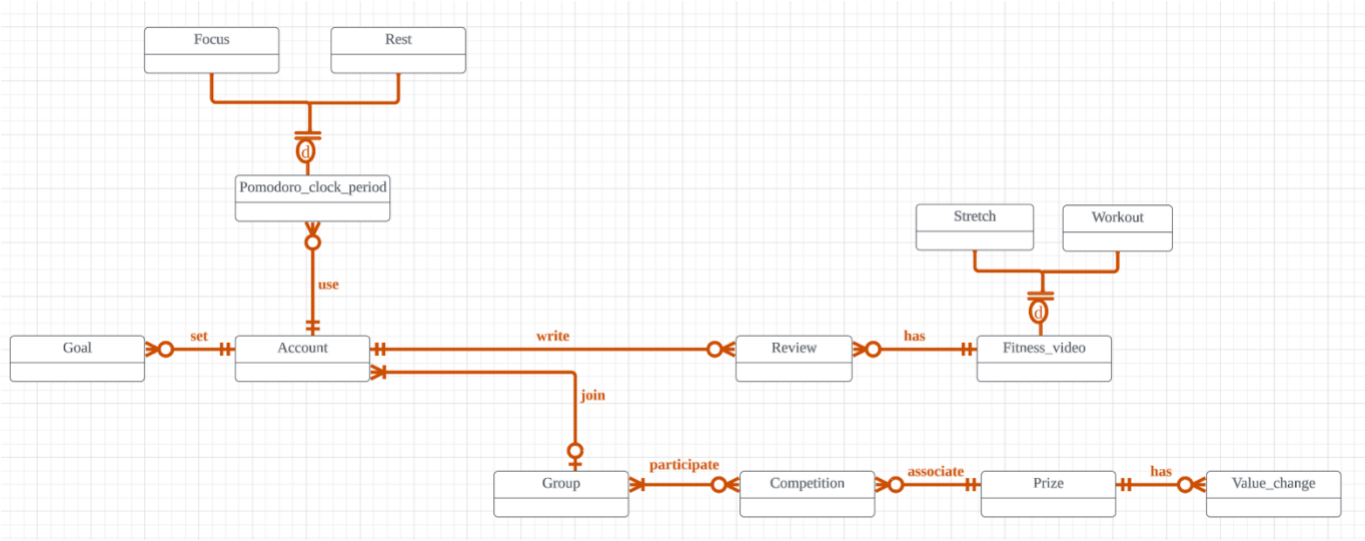
As far as the three query driven indexes, Match.winner and Focus.is_one_pomo are Boolean type, they are in the where clause. So I create indexes on them. It's obvious that they are not unique. It's reasonable that the date of Goal will be a limiting column in queries, because analysts will commonly want to limit their analysis by date range, such as a particular year, quarter, month, or week. So I select Goal.date to index. This would be a non-unique index because many goals can be set on the same day.
The end time of Period is translated to Date type in queries, so it's unnecessary to index.
Lastly, it's reasonable that the name of Account/AccountGroup/Competition/Prize will be a limiting column for some queries, a lot of subqueries use these names to look up. It's common for user to get the information using their user name, or get the information about groups, competitions and prizes with their names. So I select Account.account_name, AccountGroup.group_name, Competition.competition_name and Prize.prize_name for an index. They are all unique, because the name is unique in their table.

```sql
--INDEXES
--Replace this with your index creations.

CREATE INDEX period_account_id_index ON Period(account_id);
CREATE INDEX goal_account_id_index ON Goal(account_id);
CREATE INDEX account_group_id_index ON Account(group_id);
CREATE INDEX review_account_id_index ON Review(account_id);
CREATE INDEX review_vedio_id_index ON Review(vedio_id);
CREATE INDEX match_competition_id_index ON Match(competition_id);
CREATE INDEX competition_prize_id_index ON Competition(prize_id);
CREATE INDEX goal_date_index ON Goal(date);
CREATE INDEX match_winner_index ON Match(winner);
CREATE INDEX focus_is_one_pomo_index ON Focus(is_one_pomo);
CREATE UNIQUE INDEX account_account_name_index ON Account(account_name);
CREATE UNIQUE INDEX accountgroup_group_name_index ON AccountGroup(group_name);
CREATE UNIQUE INDEX prize_prize_name_index ON Prize(prize_name);
CREATE UNIQUE INDEX competition_competition_name_index ON Competition(competition_name);

```

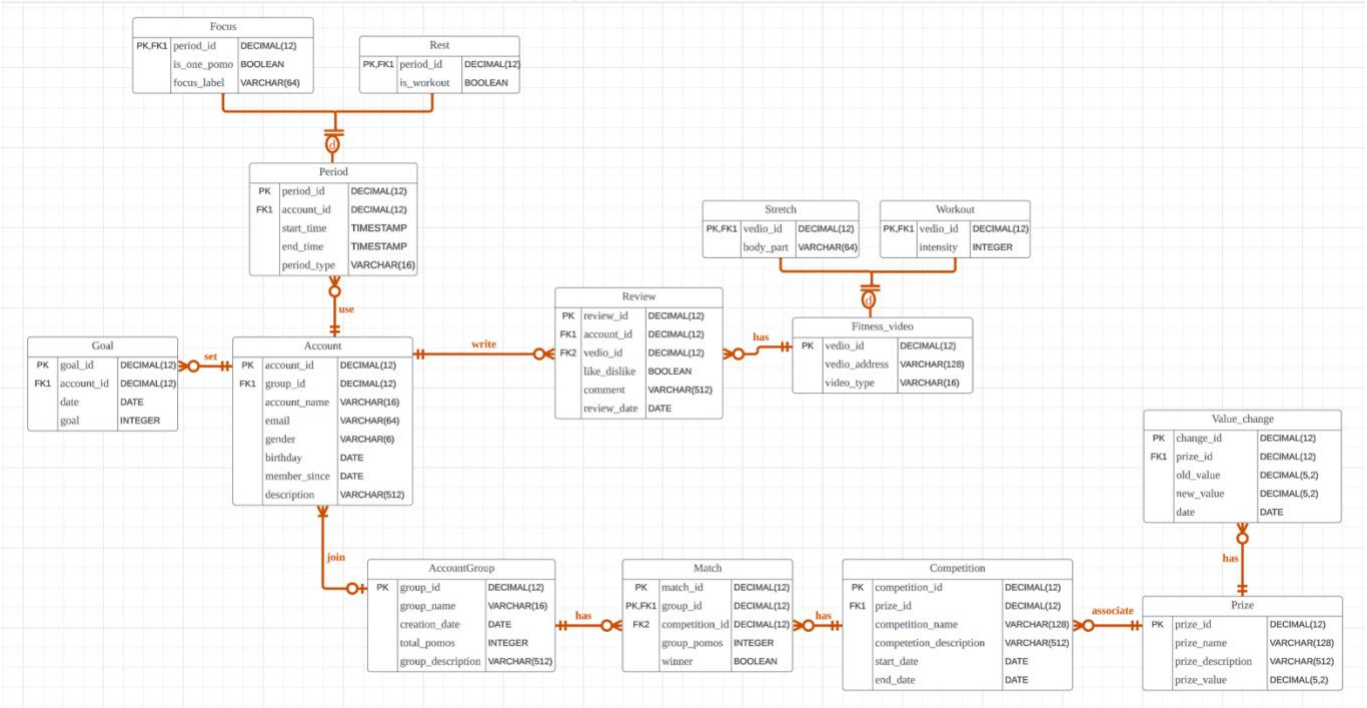
History Table Demonstration

In reviewing my DBMS physical ERD, one piece of data that would obviously benefit from a historical record the prize value in the Prize table. Such a history would help me to check the operation cost of this app accurate over time. First, my new structural database rule is: A prize may have different value changes over time; each value change must be related to one prize.

My updated conceptual ERD is below.



I added the Value_change entity and related it to Prize. My updated DBMS physical ERD is below.



The Value_change entity is present and linked to Prize. Below are the attributes I added and why.

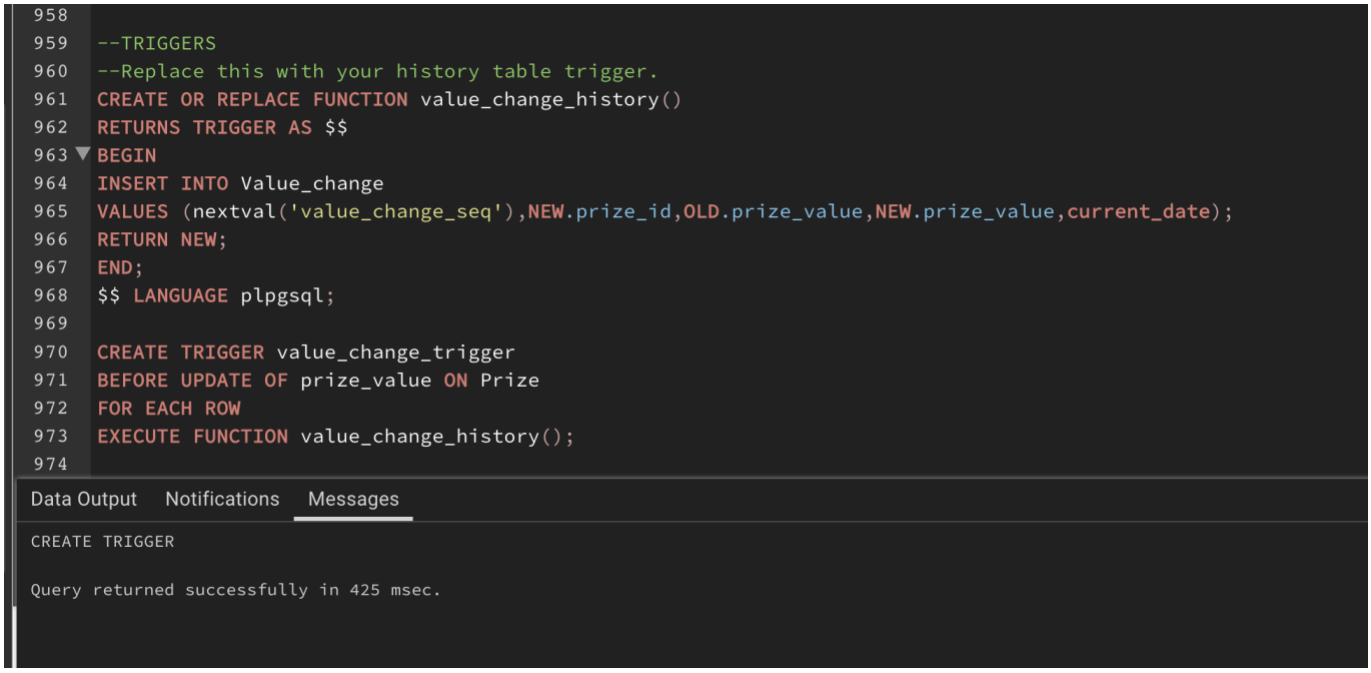
Attribute	Description	Example
change_id	A unique identifier for the change record.	1
prize_id	To indicate which prize has the value change.	1
old_value	The old value of the prize, so the data type is the same as the prize_value	20
new_value	The new value of the prize, so the data type is the same as the prize_value	25
date	The date of current date	2023-10-15

Here is a screenshot of my table and sequence creation, which has all of the same attributes and datatypes as indicated in the DBMS physical ERD.

```
126  CREATE TABLE Value_change(
127  change_id DECIMAL(12) NOT NULL PRIMARY KEY,
128  prize_id DECIMAL(12) NOT NULL REFERENCES Prize(prize_id),
129  old_value DECIMAL(5,2) NOT NULL,
130  new_value DECIMAL(5,2) NOT NULL,
131  date DATE NOT NULL
132  );
```

```
146  CREATE SEQUENCE value_change_seq;
```

Here is a screenshot of my trigger creation which will maintain the Value_change table.



```
958
959  --TRIGGERS
960  --Replace this with your history table trigger.
961  CREATE OR REPLACE FUNCTION value_change_history()
962  RETURNS TRIGGER AS $$
963  BEGIN
964  INSERT INTO Value_change
965  VALUES (nextval('value_change_seq'),NEW.prize_id,OLD.prize_value,NEW.prize_value,current_date);
966  RETURN NEW;
967  END;
968  $$ LANGUAGE plpgsql;
969
970  CREATE TRIGGER value_change_trigger
971  BEFORE UPDATE OF prize_value ON Prize
972  FOR EACH ROW
973  EXECUTE FUNCTION value_change_history();
974
```

Data Output | Notifications | Messages

```
CREATE TRIGGER

Query returned successfully in 425 msec.
```

I explain it here line by line.

CODE	DESCRIPTION
CREATE OR REPLACE FUNCTION value_change_history()	This statement creates a function named value_change_history. This function appears to be designed to capture the change history of the prize_value column in the Prize table.
RETURNS TRIGGER AS $$	This indicates that the function will return a trigger object.
BEGIN	This marks the beginning of the function body, signifying that the following code block is the body of the function.
INSERT INTO Value_change VALUES (nextval('value_change_seq'), NEW.prize_id,OLD.prize_value,NEW.prize_value,current_date);	insert the record into the Value_change table.The primary key is set by using the value_change_seq. The old and new values and prize_id are used from the variables. The date of the change is obtained by using the built-in current_date.
RETURN NEW;	This indicates that the function will return the new data row in the trigger.
END;	This marks the end of the function body.
$$ LANGUAGE plpgsql;	This indicates that the programming language used for the function is PL/pgSQL.
CREATE TRIGGER value_change_trigger	This statement creates a trigger named value_change_trigger
BEFORE UPDATE OF prize_value ON Prize	This signifies that the trigger will fire before an update to the prize_value column in the Prize table.

FOR EACH ROW	This indicates that the trigger will fire for each individual row of data.
EXECUTE FUNCTION value_change_history()	This statement means that the trigger will execute the previously defined function named value_change_history.

I start by ensuring there are prizes created. In this case, there is one has an ID of 1 called 'Coffee Gift Card' with a value of 25 as illustrated by the screenshot below.

```
974
975  SELECT * FROM Prize;
976
```

Data Output | Notifications | Messages

	prize_id [PK] numeric (12)	prize_name character varying (128)	prize_description character varying (512)	prize_value numeric (5,2)
1	1	Coffee Gift Card	A gift card for a popular coffee shop	25.00
2	2	Bookstore Voucher	A voucher for a local bookstore	40.00
3	3	Movie Tickets	Tickets for a night out at the movies	30.00
4	4	Gift Card	A $25 gift card	25.00
5	5	T-Shirt	Official event T-shirt	15.00

Next, I update the value to 20. Last, I verify that the Value_change table has a record of this value change in the screenshot below.

```
976
977  UPDATE Prize
978  SET prize_value=20
979  WHERE prize_name='Coffee Gift Card';
980
981  SELECT * FROM Value_change;
```

Data Output | Notifications | Messages

	change_id [PK] numeric (12)	prize_id numeric (12)	old_value numeric (5,2)	new_value numeric (5,2)	date date
1	1	1	25.00	20.00	2023-10-15

Data Visualizations

This app is designed to help users train their attention and improve their time utilization , it is important to give the users a weekly report to show their performance in this week.

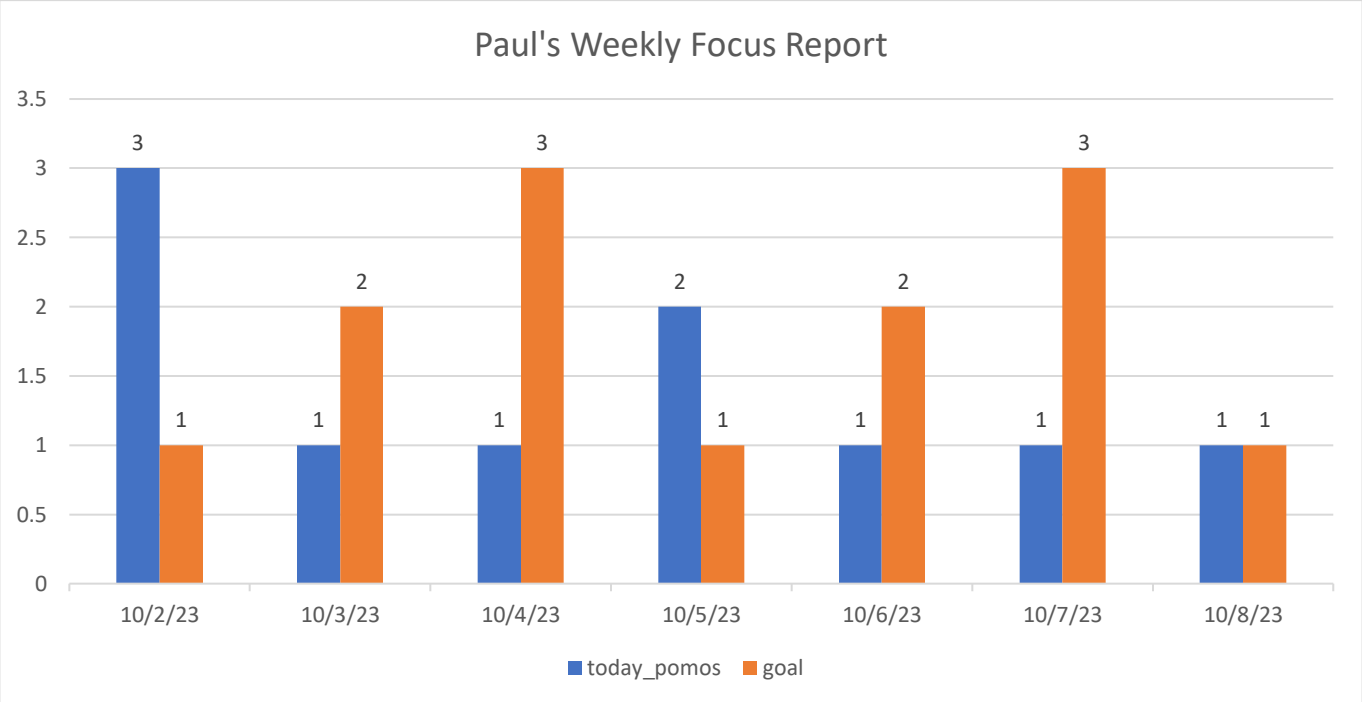
I develop and execute a query that shows the performance of the user in this week, how many pomos they have completed every day and whether they have achieved their goal. The result is ordered by the date. This query and results are shown below.

```
1034  SELECT * FROM GoalAchievement
1035  WHERE account_name='Paul' AND date<='2023-10-08' AND date>='2023-10-02'
1036  ORDER BY date;
1037
```

Data Output | Notifications | Messages

	account_name character varying (16)	goal integer	date date	today_pomos bigint	goal_achieved text
1	Paul	1	2023-10-02	3	Yes
2	Paul	2	2023-10-03	1	No
3	Paul	3	2023-10-04	1	No
4	Paul	1	2023-10-05	2	Yes
5	Paul	2	2023-10-06	1	No
6	Paul	3	2023-10-07	1	No
7	Paul	1	2023-10-08	1	Yes

Since these results only have two measures, I plan to use a simple bar chart to visualize the results. After exporting to a CSV and creating the bar chart in Excel, I see the following result.



I can tell several things from this visualization. Paul insists using this app to help him focus every day. However, he only completed his goal in three days, which indicates that he wants to have more focusing periods but he is unable to do that. Perhaps he needs to lower his goals and gradually improve his focus.

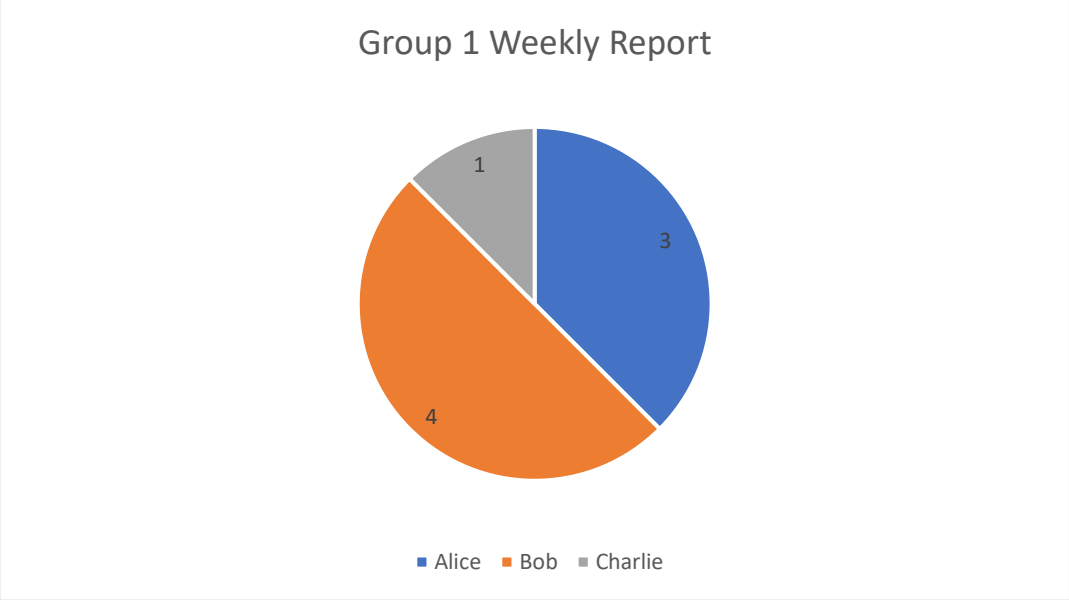
For a group, every group members' effort matters. If members achieved more pomos, their group's total pomos will be more and they may win a match if they participate a competition. It's necessary to show a weekly report on every group members' contribution to the group's total pomos. This query and results are shown below.

```
1048  SELECT a.account_name,a.group_id,COUNT(*) AS weekly_pomos
1049  FROM Focus AS f
1050  LEFT JOIN Period AS p ON f.period_id = p.period_id
1051  JOIN Account AS a ON p.account_id = a.account_id
1052  WHERE f.is_one_pomo = true AND
1053        DATE(p.end_time)<='2023-10-08' AND DATE(p.end_time)>='2023-10-02' AND
1054        a.group_id=1
1055  GROUP BY a.account_name, a.group_id
```

Data Output | Notifications | Messages

	account_name character varying (16)	group_id numeric (12)	weekly_pomos bigint
1	Alice	1	3
2	Bob	1	4
3	Charlie	1	1

Since these results only have one measure, I plan to use a simple pie chart to visualize the results. After exporting to a CSV and creating the pie chart in Excel, I see the following result.



I can tell several things from this visualization. The group1's members are Alice, Bob and Charlie. It's clearly that Bob completed most pomos in this week, Alice comes to the second and Alice finished in second place and her pomos' quantity is almost the same as Bob's, Charlie finished the least. Alice and Bob may encourage Charlie to make more effort on focusing.

Summary and Reflection

After working hard for six weeks, I've got whole database for my Pomodoro-Workout Clock ready, and I'm really happy about it. My database helps users training their focus ability and increase their time utilization.

The structural database rules and conceptual ERD for my database design contain the important entities of Account, Period, AccountGroup and other entities, as well as relationships between them. The design contains a hierarchy of Period/Focus and Period/Rest to reflect the two primary ways people use their time. The design also contains a hierarchy of Fitness_video/Stretch and Fitness_video/Workout to reflect two different types of fitness video. The DBMS physical ERD contains the same entities and relationships, uses the best practice of synthetic keys, and contains the important attributes needed by the database to support the application.

The SQL script that contains all table creations that follows the specification from the DBMS physical ERD exactly. Important indexes have been created to help speed up access to my database and are also available in an index script. Stored procedures have been created and executed transactionally to populate some of my database with data. Some questions useful to Pomodoro-Workout Clock have been identified, and implemented with SQL queries. Some useful visualizations have been presented, along with the stories they tell.

As I reflect on the database and my accomplishments, I can say it’s been a long but rewarding road. It is amazing to see a real database in action that can be hooked up to the Pomodoro-Workout Clock. I can envision many of the screens already. I can see there is still more to develop in my database, but feel it’s a solid foundation to move forward with.