Online Coaching Application

In the fitness and coaching domain, various challenges can impede the effectiveness of traditional coaching methods. Identifying these challenges helps in crafting a solution that caters specifically to the needs of gym coaching. Some key problems include:

1. **Lack of Personalization**: Traditional gym coaching may struggle to provide personalized workout plans tailored to individual fitness goals, leading to suboptimal results for clients with diverse needs.

2. **Inefficient Progress Tracking**: Monitoring client progress and adjusting workout plans accordingly can be cumbersome using manual methods. This may result in a lack of timely adjustments and hinder clients from achieving their fitness objectives.

3. **Limited Access to Resources**: Clients might face challenges accessing relevant fitness resources and information to support their training outside coaching sessions. This lack of accessibility can impact their ability to adhere to a holistic fitness regimen.

4. **Communication Barriers**: Effective communication between coaches and clients is crucial for addressing concerns, providing guidance, and maintaining motivation. Traditional communication channels may not be sufficiently streamlined for this purpose.

Our gym coaching application creates a virtual fitness world where users receive personalized workout plans tailored to their fitness goals, considering their fitness level, preferences, and health considerations. The app ensures a seamless progress tracking system, enabling both clients and coaches to monitor achievements, make informed adjustments, and celebrate milestones.

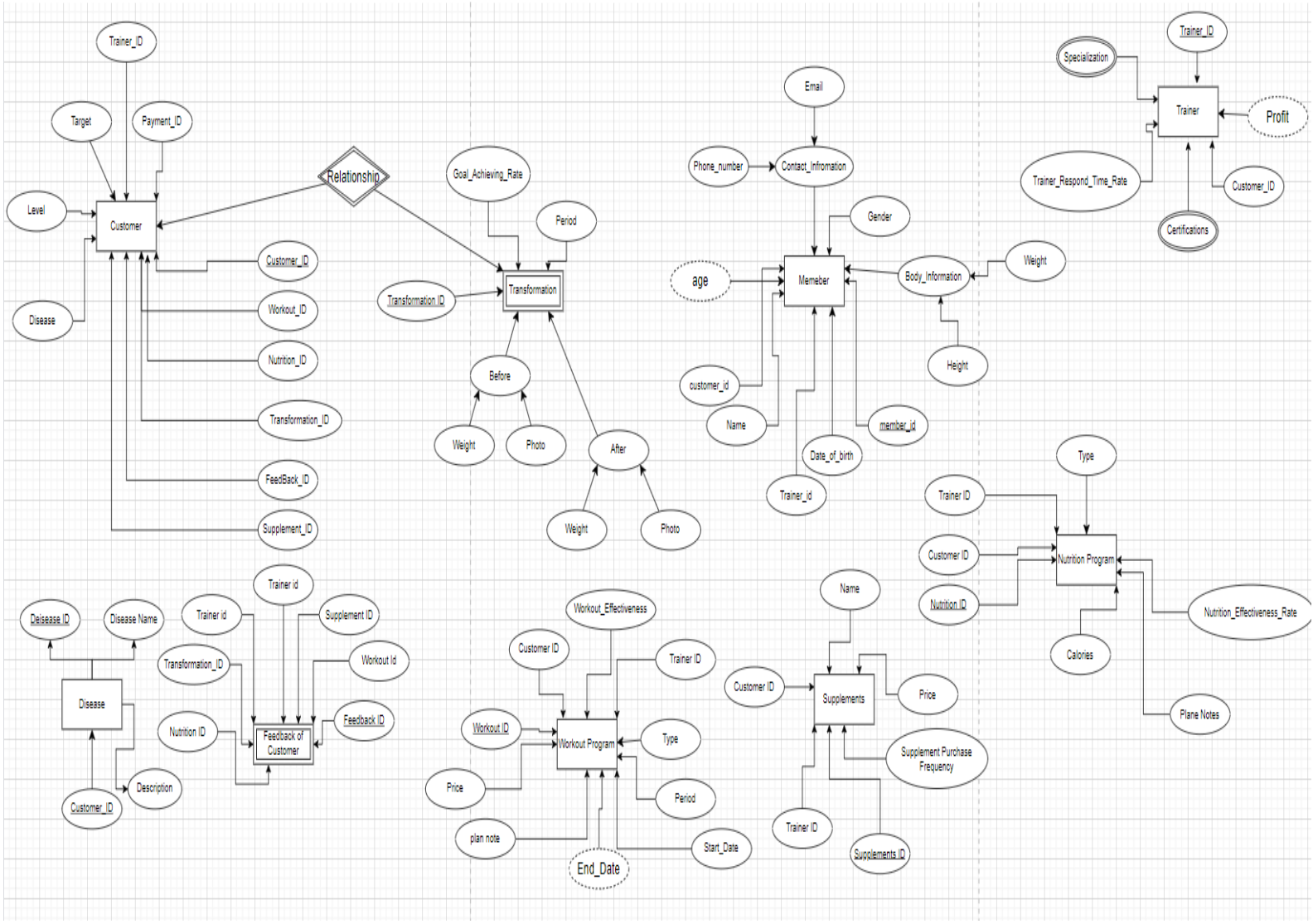
To enrich the fitness experience, the app provides a diverse resource library featuring nutrition guides and practical fitness tips. This not only supports users during coaching sessions but also empowers them with valuable information for their overall well-being.

In essence, our gym coaching app transforms the coaching experience, creating a nurturing and personalized environment that empowers users on their fitness journey.

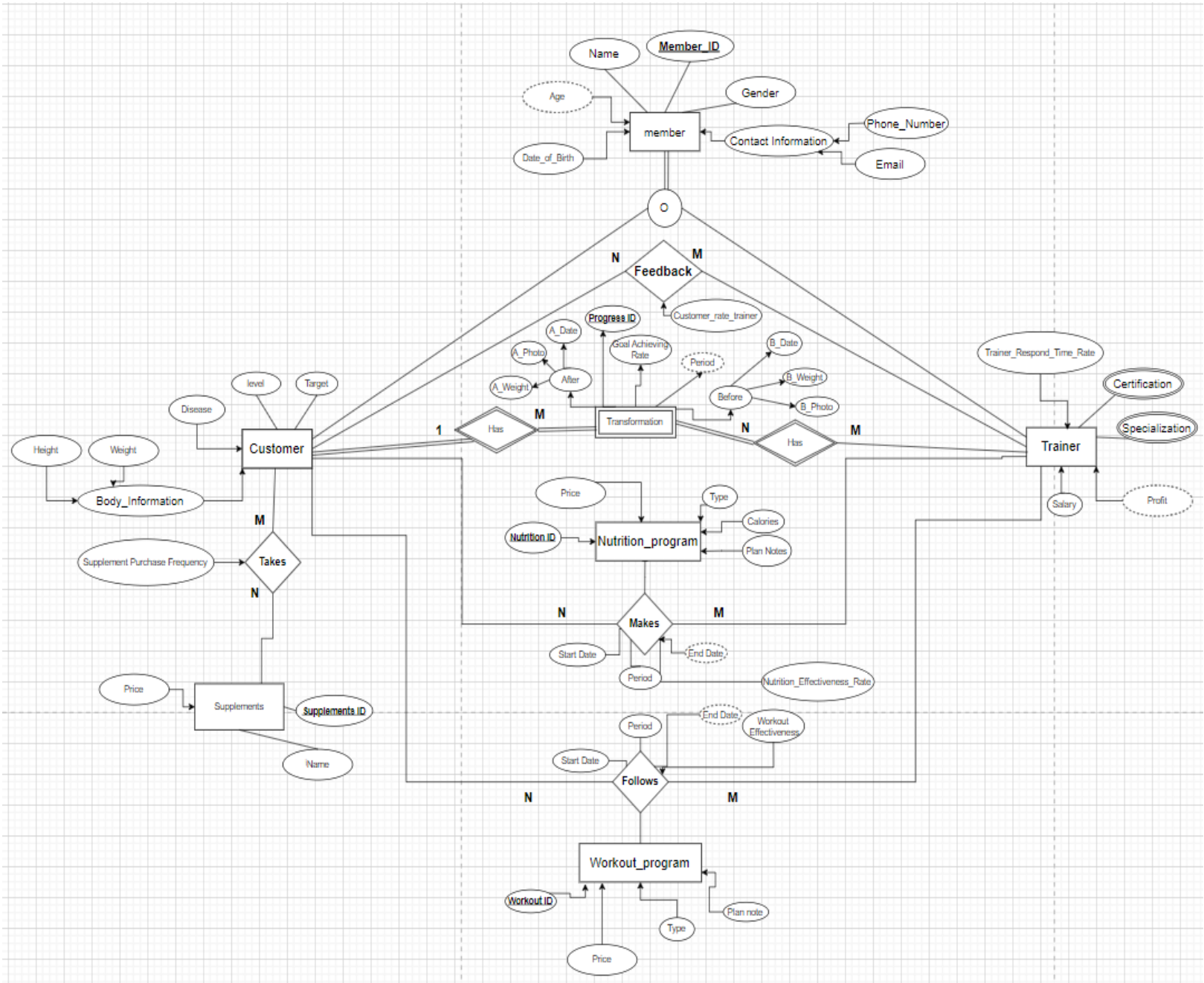
Reference:

NSCA - National Strength and Conditioning Association. (2020). "NSCA's Essentials of Personal Training." Human Kinetics.

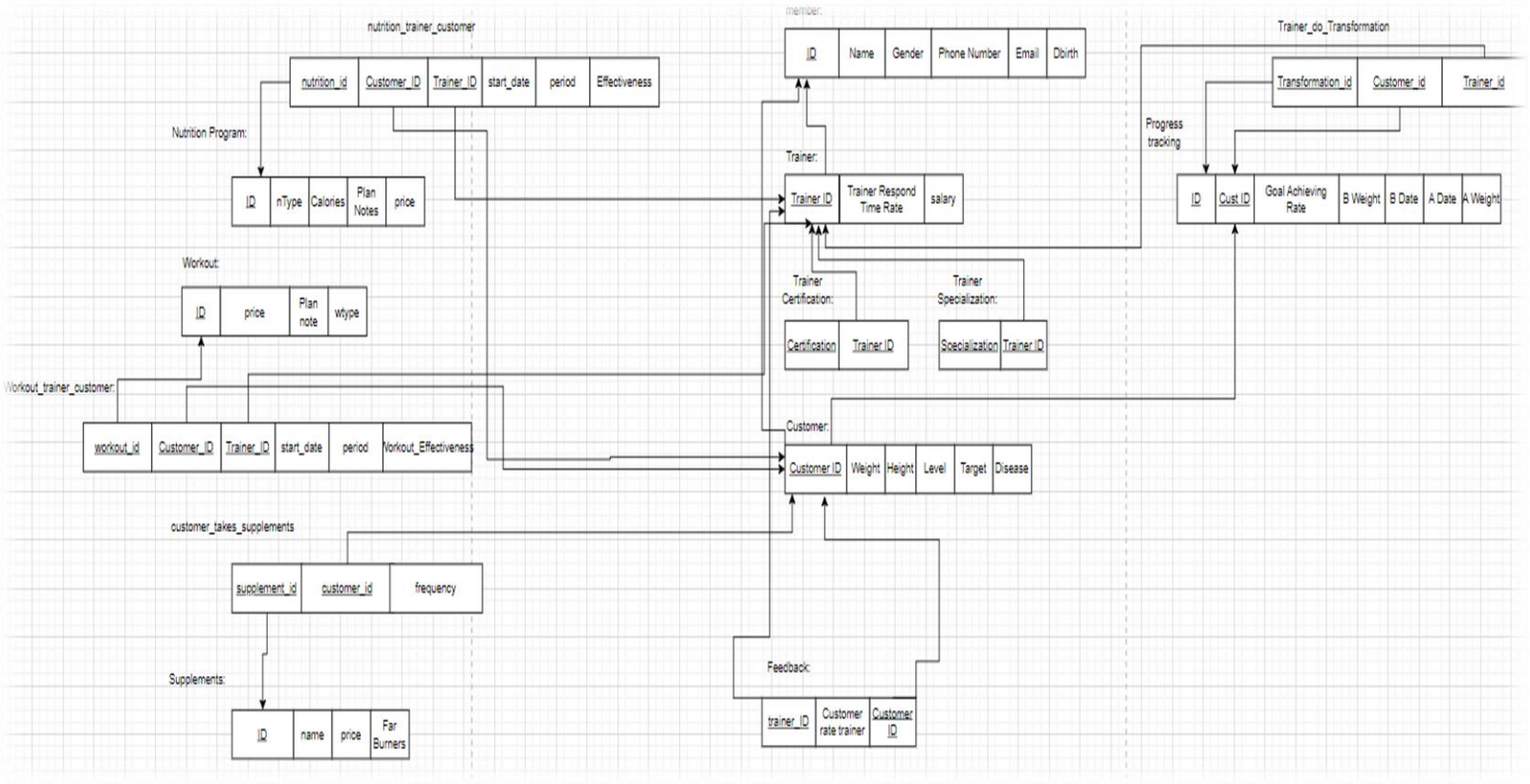
Design rough ER schema



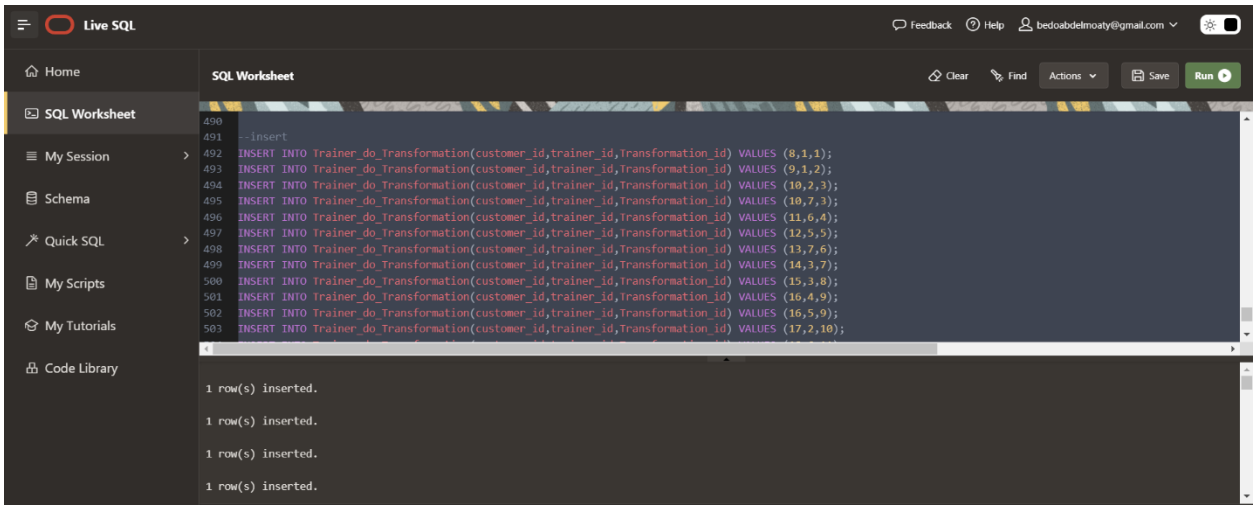
Design the final ER diagram.

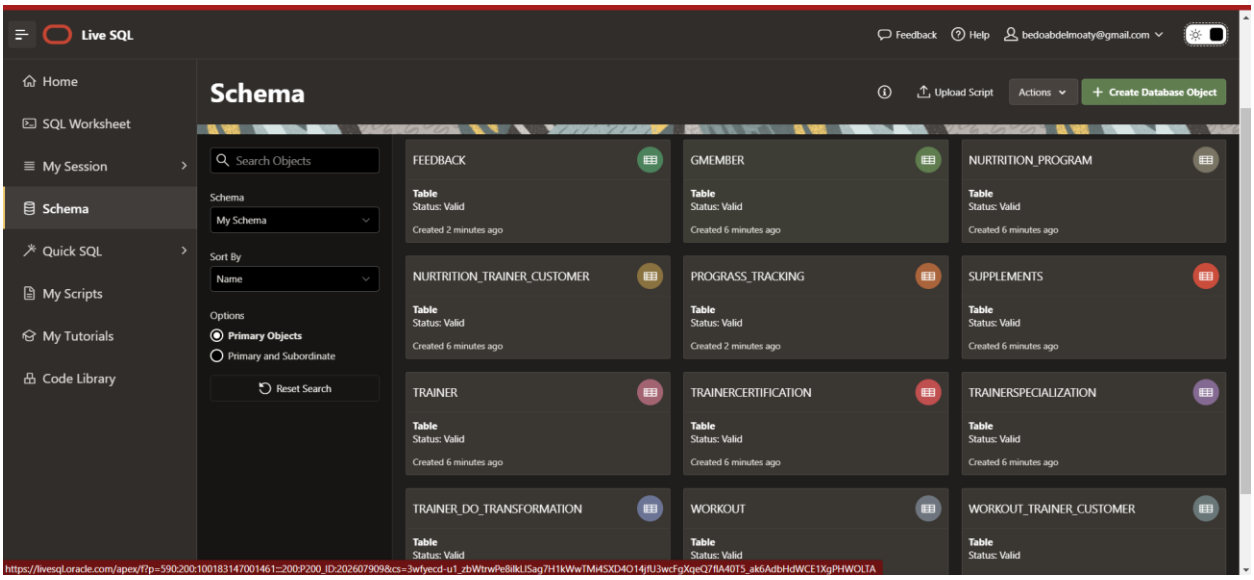


Relational mapping



DDL in Oracle (with insertion) samples





Complex Queries

1) **Retrieve display names and id that trainer is customer.**

1st Query:

```
create view customer1 as select * from customer natural join gmember;
create view trainer1 as select * from trainer natural join gmember;
select id, name from trainer1 natural join customer1 order by id;
```

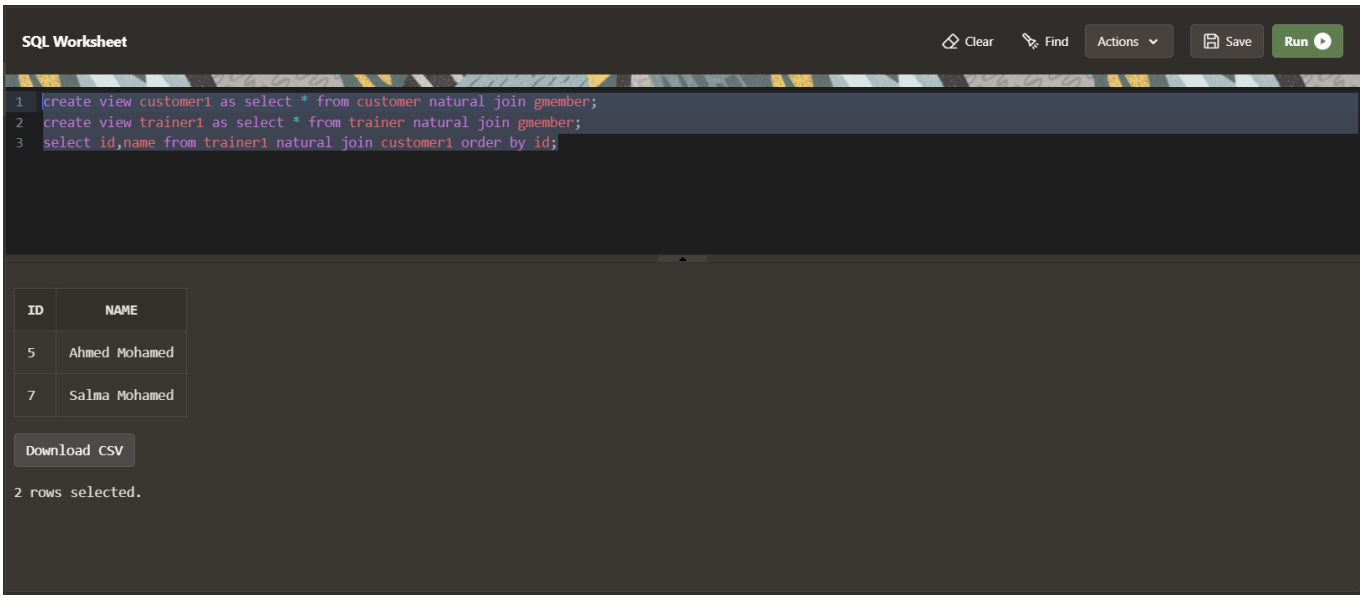
Relation Algebra:

$$customer1 \leftarrow \pi_{*}(customer * gmember)$$

$$trainer1 \leftarrow \pi_{*}(trainer * gmember)$$

$$R \leftarrow \pi_{id,\ name}(trainer1 * customer1)$$

$$R_{ordered} \leftarrow \rho_{id \rightarrow id\_ordered}(R)$$



2) Retrieve display names and rates for customer feedback trainer

2nd Query:

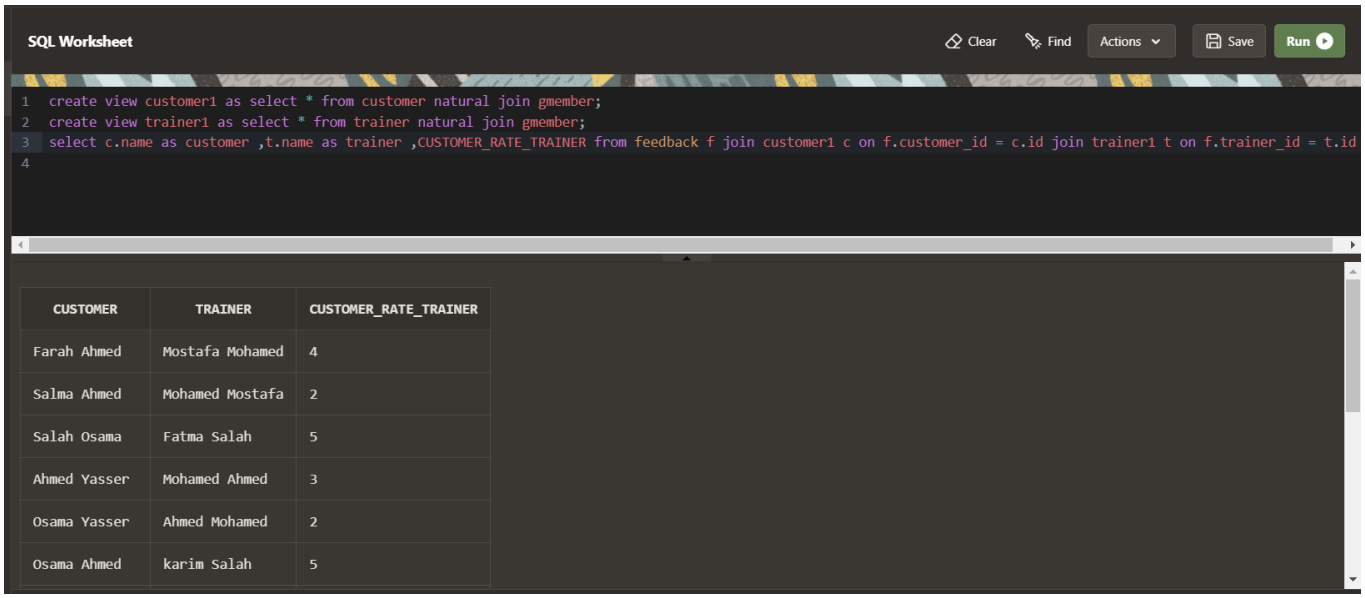
```
create view customer1 as select * from customer natural join gmember;
create view trainer1 as select * from trainer natural join gmember;
select c.name as customer ,t.name as trainer ,CUSTOMER_RATE_TRAINER from feedback f
join customer1 c on f.customer_id = c.id join trainer1 t on f.trainer_id = t.id ;
```

Relation Algebra:

$$customer1 \leftarrow \pi_{*}(customer * gmember)$$

$$trainer1 \leftarrow \pi_{*}(trainer * gmember)$$

$$R \leftarrow \pi_{c.name,t.name,CUSTOMER\_RATE\_TRAINER}(\sigma_{f.customer\_id=c.id \wedge f.trainer\_id=t.id}(f \bowtie_{f.customer\_id=c.id} c \bowtie_{f.trainer\_id=t.id} t))$$



CUSTOMER	TRAINER	CUSTOMER_RATE_TRAINER
Farah Ahmed	Mostafa Mohamed	4
Salma Ahmed	Mohamed Mostafa	2
Salah Osama	Fatma Salah	5
Ahmed Yasser	Mohamed Ahmed	3
Osama Yasser	Ahmed Mohamed	2
Osama Ahmed	karim Salah	5

3) **Retrieve display customer's name, supplement's name , price and FREQUENCY that customer take supplements**

3rd Query:

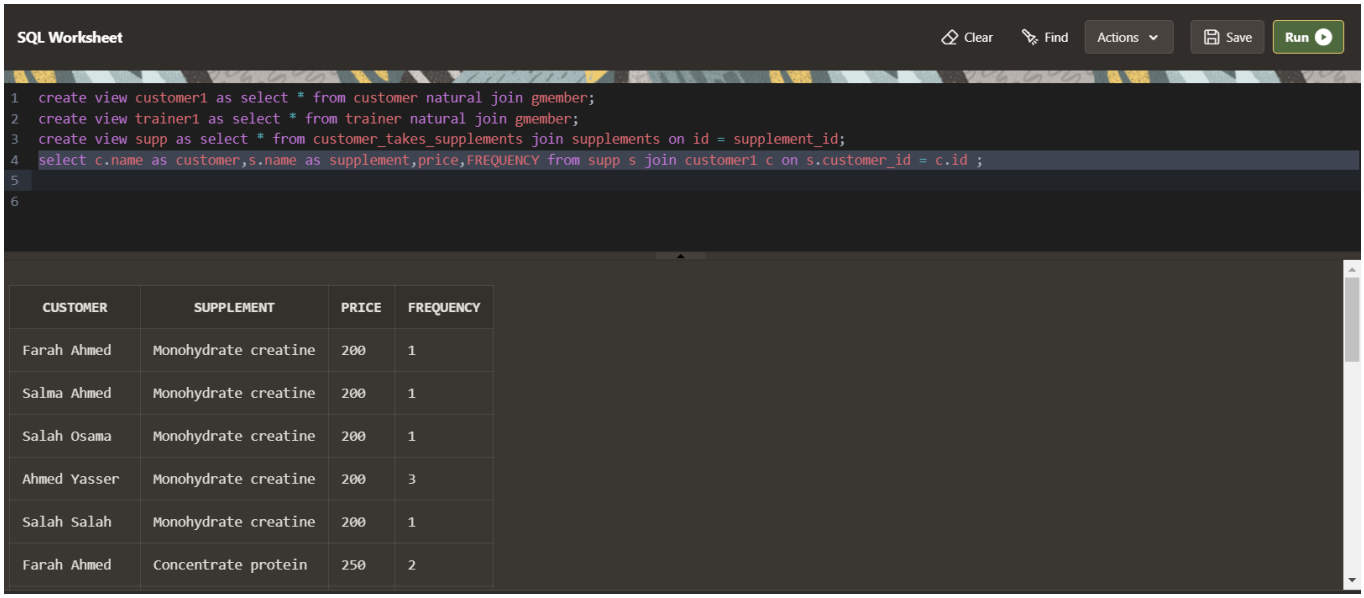
```
create view customer1 as select * from customer natural join gmember;
create view supp as select * from customer_takes_supplements join supplements on id = supplement_id;
select c.name as customer, s.name as supplement, price, FREQUENCY from supp s join customer1 c on s.customer_id = c.id ;
```

Relation Algebra:

$$customer1 \leftarrow \pi_{*}(customer * gmember)$$

$$supp \leftarrow \pi_{*}(customer\_takes\_supplements \bowtie_{id=supplement\_id} supplements)$$

$$R \leftarrow \pi_{c.name,\ s.name,\ \text{price},\ \text{FREQUENCY}}(\sigma_{s.customer\_id=c.id}(S \bowtie_{s.customer\_id=c.id} C))$$



CUSTOMER	SUPPLEMENT	PRICE	FREQUENCY
Farah Ahmed	Monohydrate creatine	200	1
Salma Ahmed	Monohydrate creatine	200	1
Salah Osama	Monohydrate creatine	200	1
Ahmed Yasser	Monohydrate creatine	200	3
Salah Salah	Monohydrate creatine	200	1
Farah Ahmed	Concentrate protein	250	2

4) **Retrieve display customer's name, trainer's name , type , EFFECTIVENESS , CALORIES and start date and end date for customer take nutrition**

4th Query:

```
create view customer1 as select * from customer natural join gmember;
create view trainer1 as select * from trainer natural join gmember;
create view nurtrition1 as select * from nurtrition_trainer_customer join
nurtrition_program on id = nurtrition_id;
select TRAINER_ID , t.name as trainer ,CUSTOMER_ID ,c.name as customer ,
ntype,CALORIES ,EFFECTIVENESS,start_date,start_date+period as end_date from
nurtrition1 n join customer1 c on c.id = n.customer_id join trainer1 t on t.id =
n.trainer_id ;
```

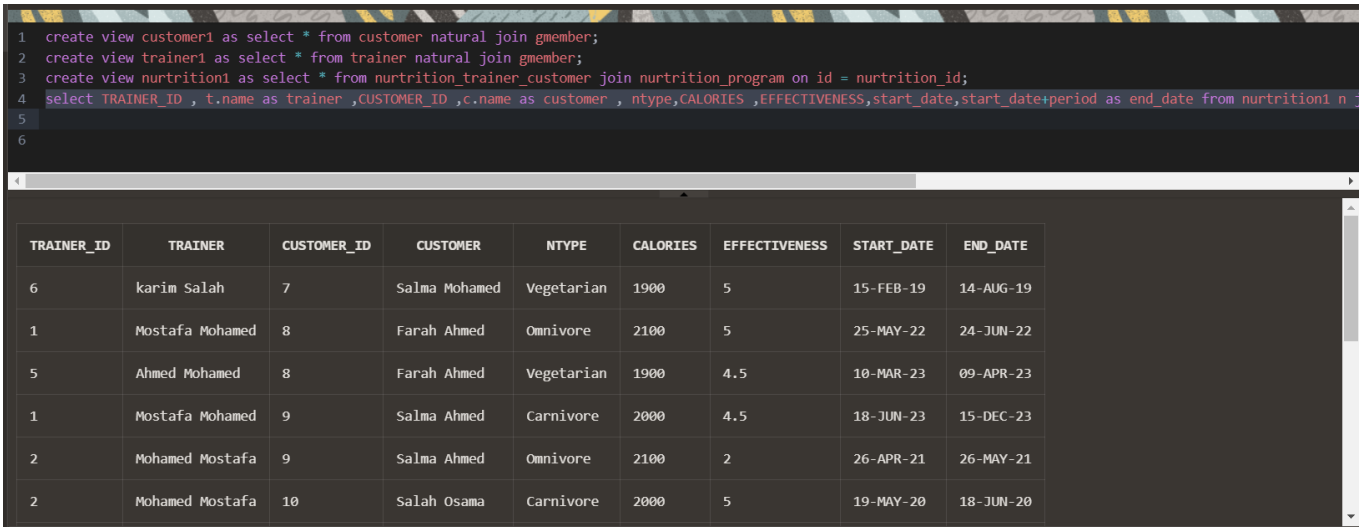
Relation Algebra:

$$customer1 \leftarrow \pi_{*}(customer * gmember)$$

$$trainer1 \leftarrow \pi_{*}(trainer * gmember)$$

$$nurtrition1 \leftarrow \pi^{*}(nurtrition\_trainer\_customer \bowtie_{id=nurtrition\_id} nurtrition\_program)$$

$$R \leftarrow \pi_{TRAINER\_ID,t.name,CUSTOMER\_ID,c.name,ntype,CALORIES,price,EFFECTIVENESS,start\_date,start\_date+period\ as\ end\_date}\left(n \bowtie_{c.id=n.customer\_id} c \bowtie_{t.id=n.trainer\_id} t\right)$$



```
create view customer1 as select * from customer natural join gmember;
create view trainer1 as select * from trainer natural join gmember;
create view nurtrition1 as select * from nurtrition_trainer_customer join nurtrition_program on id = nurtrition_id;
select TRAINER_ID , t.name as trainer ,CUSTOMER_ID ,c.name as customer , ntype,CALORIES ,EFFECTIVENESS,start_date,start_date+period as end_date from nurtrition1 n
```

TRAINER_ID	TRAINER	CUSTOMER_ID	CUSTOMER	NTYPE	CALORIES	EFFECTIVENESS	START_DATE	END_DATE
6	karim Salah	7	Salma Mohamed	Vegetarian	1900	5	15-FEB-19	14-AUG-19
1	Mostafa Mohamed	8	Farah Ahmed	Omnivore	2100	5	25-MAY-22	24-JUN-22
5	Ahmed Mohamed	8	Farah Ahmed	Vegetarian	1900	4.5	10-MAR-23	09-APR-23
1	Mostafa Mohamed	9	Salma Ahmed	Carnivore	2000	4.5	18-JUN-23	15-DEC-23
2	Mohamed Mostafa	9	Salma Ahmed	Omnivore	2100	2	26-APR-21	26-MAY-21
2	Mohamed Mostafa	10	Salah Osama	Carnivore	2000	5	19-MAY-20	18-JUN-20

5) **Retrieve display customer's name, trainer's name , type , start date and end date for customer do workout**

5th Query:

```
create view customer1 as select * from customer natural join gmember;
create view trainer1 as select * from trainer natural join gmember;
create view workout1 as select * from workout join workout_trainer_customer on id = workout_id;
select customer_id, c.name as customer,trainer_id,t.name as trainer,wtype,price,start_date,start_date+period as end_date from (workout1 w join customer1 c on c.id = w.CUSTOMER_ID) join trainer1 t on t.id = w.trainer_id ;
```

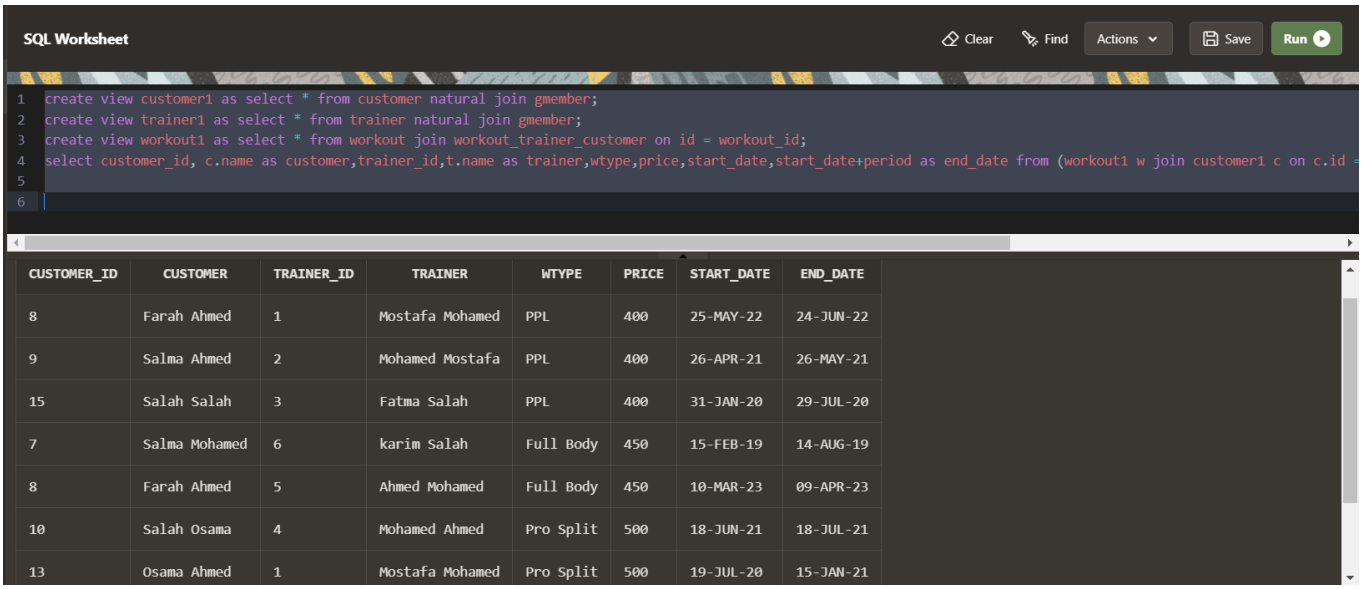
Relation Algebra:

$$customer1 \leftarrow \pi_{*}(customer * gmember)$$

$$trainer1 \leftarrow \pi_{*}(trainer * gmember)$$

$$workout1 \leftarrow \pi_{*}(workout \bowtie_{\text{id=workout\_id}} \text{workout\_trainer\_customer})$$

$$R \leftarrow \pi_{\text{customer\_id},c.\text{name},\text{trainer\_id},t.\text{name},\text{wtype},\text{price},\text{start\_date},\text{start\_date+period as end\_date}} (w \bowtie_{c.\text{id}=w.\text{CUSTOMER\_ID}} c \bowtie_{t.\text{id}=w.\text{trainer\_id}} t$$



CUSTOMER_ID	CUSTOMER	TRAINER_ID	TRAINER	WTYPE	PRICE	START_DATE	END_DATE
8	Farah Ahmed	1	Mostafa Mohamed	PPL	400	25-MAY-22	24-JUN-22
9	Salma Ahmed	2	Mohamed Mostafa	PPL	400	26-APR-21	26-MAY-21
15	Salah Salah	3	Fatma Salah	PPL	400	31-JAN-20	29-JUL-20
7	Salma Mohamed	6	karim Salah	Full Body	450	15-FEB-19	14-AUG-19
8	Farah Ahmed	5	Ahmed Mohamed	Full Body	450	10-MAR-23	09-APR-23
10	Salah Osama	4	Mohamed Ahmed	Pro Split	500	18-JUN-21	18-JUL-21
13	Osama Ahmed	1	Mostafa Mohamed	Pro Split	500	19-JUL-20	15-JAN-21

6) Retrieve all customers that subscribed All subscription

6th Query:

```
create view customer1 as select * from customer natural join gmember;
create view workout1 as select * from workout join workout_trainer_customer on id = workout_id;
create view supp as select * from customer_takes_supplements join supplements on id = supplement_id;
create view nurtrition1 as select * from nurtrition_trainer_customer join nurtrition_program on id = nurtrition_id;
select c.id , c.name as customer,s.name as supplement,w.wtype,n.ntype from ((customer1 c join  workout1 w on c.id = w.CUSTOMER_ID) join nurtrition1 n on c.id = n.CUSTOMER_ID)  join supp s on c.id = s.CUSTOMER_ID ;
```

Relation Algebra:

$customer1 \leftarrow \pi_{*}(customer * gmember)$

$workout1 \leftarrow \pi_{*}(workout \bowtie_{id=workout\_id} workout\_trainer\_customer)$

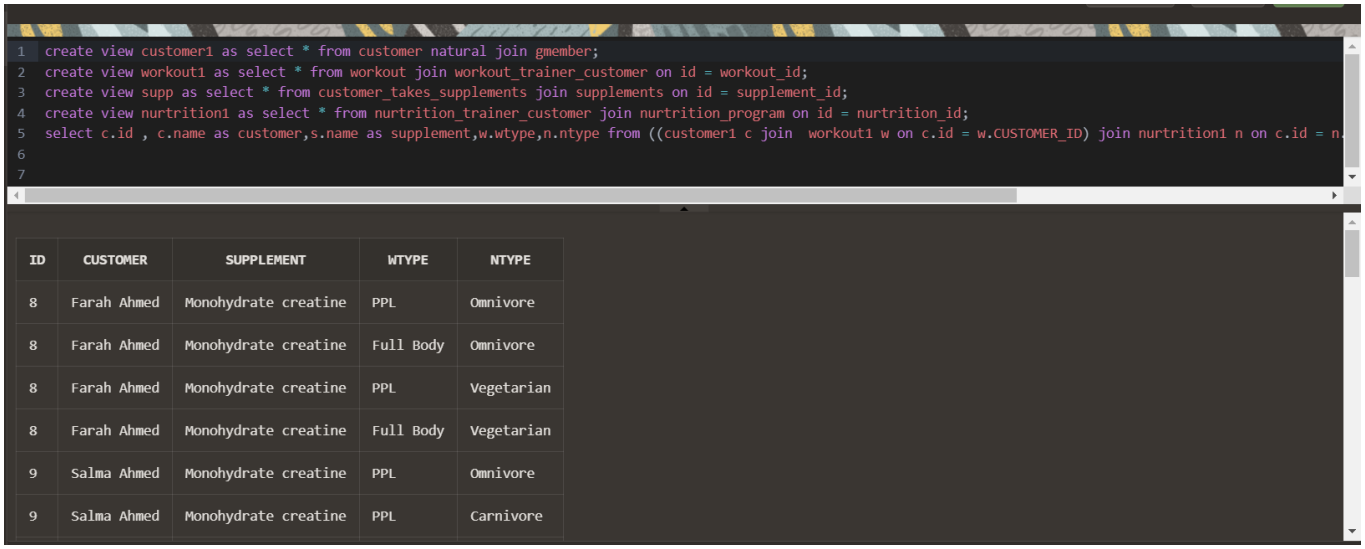
$supp \leftarrow \pi_{*}(customer\_takes\_supplements \bowtie_{id=supplement\_id} supplements)$

$nurtrition1 \leftarrow \pi_{*}(nurtrition\_trainer\_customer \bowtie_{id=nurtrition\_id} nurtrition\_program)$

$R_1 \leftarrow c \bowtie_{c.id=w.CUSTOMER\_ID} w$

$R_2 \leftarrow R_1 \bowtie_{c.id=n.CUSTOMER\_ID} n$

$R \leftarrow R_2 \bowtie_{c.id=s.CUSTOMER\_ID} s$ $R_{result} \leftarrow \pi_{c.id,c.name \rightarrow customer,s.name \rightarrow supplement,w.wtype,n.ntype}(R)$



```
create view customer1 as select * from customer natural join gmember;
create view workout1 as select * from workout join workout_trainer_customer on id = workout_id;
create view supp as select * from customer_takes_supplements join supplements on id = supplement_id;
create view nurtrition1 as select * from nurtrition_trainer_customer join nurtrition_program on id = nurtrition_id;
select c.id , c.name as customer,s.name as supplement,w.wtype,n.ntype from ((customer1 c join  workout1 w on c.id = w.CUSTOMER_ID) join nurtrition1 n on c.id = n.
```

ID	CUSTOMER	SUPPLEMENT	WTYPE	NTYPE
8	Farah Ahmed	Monohydrate creatine	PPL	Omnivore
8	Farah Ahmed	Monohydrate creatine	Full Body	Omnivore
8	Farah Ahmed	Monohydrate creatine	PPL	Vegetarian
8	Farah Ahmed	Monohydrate creatine	Full Body	Vegetarian
9	Salma Ahmed	Monohydrate creatine	PPL	Omnivore
9	Salma Ahmed	Monohydrate creatine	PPL	Carnivore

Procedure & Trigger

1) Function to calculate age

```
CREATE OR REPLACE FUNCTION calculate_age(birthdate DATE)
RETURN NUMBER
IS
    age NUMBER;
BEGIN
    age := TRUNC(MONTHS_BETWEEN(SYSDATE, birthdate) / 12);
    RETURN age;
END calculate_age;
```

2) Function to get period

```
create or replace function get_period(start_date date,end_date date)
return int
is
period int;
begin
    period := end_date - start_date ;
    return period;
end get_period;
```

3) Trigger to check start date

```
create or replace trigger check_Bdate
before insert or update on prograss_tracking
for each row
declare
    current_date date;
begin
    current_date := SYSDATE;
    if: new.Bdate > current_date then
        RAISE_APPLICATION_ERROR(-20001, 'Start date cannot be in the future');
    end if;
end check_BDATE;
```

4) Trigger to give the trainer bonus from workout price.

```
CREATE OR REPLACE TRIGGER update_trainer_salary_trigger_w
AFTER INSERT ON workout_trainer_customer
FOR EACH ROW
DECLARE
    current_date DATE;
    end_date DATE;
    workout_price NUMBER;
BEGIN
    current_date := SYSDATE;
    end_date := get_end_date(:NEW.start_date, :NEW.period);

    IF end_date >= current_date THEN
        SELECT price INTO workout_price
        FROM workout
        WHERE id = :NEW.workout_id;

        UPDATE trainer
        SET salary = salary + (workout_price * 0.8)
        WHERE id = :NEW.trainer_id;
    END IF;
END update_trainer_salary_trigger_w;
```