A Project Report On

"_GYM MANAGEMENT SYSTEM_"

Submitted to

Savitribai Phule Pune University

Developed By

Mr. Harshal Ravindra Kardile

In the partial fulfillment of the degree of

Bachelor of Science (Computer Science)-II Sem-II

(2019-2020)

Under the Guidance Of

Ms. J.D.Shendge

Pune District Educations Association's

Department of Computer Science Baburaoji Gholap College, Sangvi, Pune-411027.

P.D.E.A's
Baburaoji Gholap College, Sangvi, Pune - 27.
Department of Computer Science

CERTIFICATE

Savitribai Phule Pune University

This is to certify that,Mr. Harshal Ravindra kardile Class- SY Bsc(CS) Roll No. 36 as satisfactorily completed the project under the subject "Software Engeneering" having the title "Gym Management System". As laid down by the Savitribai Phule Pune University during the year 2019 - 2020

Seat No._____

Mrs. S.S. Chowhan

Project Guide

Head of Department

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project guide, **Ms. J.D.Shendge** , for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department **Ms. S.S,Chowhan** for encouraging and allowing us to present the project on the topic "Gym Management System "at our department premises for the partial fulfillment of the requirements leading to the award of S.Y.B.Sc(Computer Science) degree.

We take this opportunity to thank all our lecturers who have directly or indirectly helped our project. We pay our respects and love to our parents and all other family members and friends for their love and encouragement throughout our career. Last but not the least we express our thanks to our friends for their cooperation and support.

Index

➢ Problem Definition

- Problem description:-

Going to the gym everyday is one way to accomplish some goals.regular exercises carries many physical mental physical mental health benefits it boosts the brain power and sharpens the memory

In this existing system customer knows the exact location of gym and facilities provided by them. Gym have to maintain the record properly, because of the lots of Paper work it is defficult to manage record properly.

It is management system consist gym and health club database, in which we can keep record on our member and their membership and have quick communication with member it also include booking system Concession ,offer.

Our gym management system is a complete gym and looks for all our members,memberships and their activities. This system also provide the total information about machinery data of coches and trainers.

- Study of existing system(manual or computerised)

In existing system of gym management system,all entries done manually there is no online system
available. it requires,

1. More man power.
2. Needs manual entries which is time consuming.
3. Lost of previous data entries.(may occurs)

- Drawbaks of existing system:-

    1. It require lot of paperwork
    2. Maintaining records and generating records become difficults
    3. Record missing issues may occur while maintaining the records

- Scope of the proposed system:-

    The system provides proper security and reduce the manual work and easy to find gym location and browsing the plans

    1. Security of data
    2. Ensure data accuracy
    3. Found record immediately
    4. Minimize manual data entries
    5. Better service
    6. User friendly

➢ Feasibility study

- Technical feasibility:-

The technical feasibility always focuses on existing computer hardware ,software and so personnel. It needs to

consider the machine availability nature of hardware & software being used.

System is portable since it can be used on any machine & no additional hardware is required

The project backend is postgresql.

- Economical feasibility:-

It considered the cost/benefit analysis of the proposed project the benefits are always expected to over weighting the costs

Economical feasibility is helpful to find the system development cost & check whether it is justifiable that we it checks

- Operational feasibility:-

It consider the acceptability of system it checks whether sys be used if it is developed and implementation are uses of the system able to handle the system

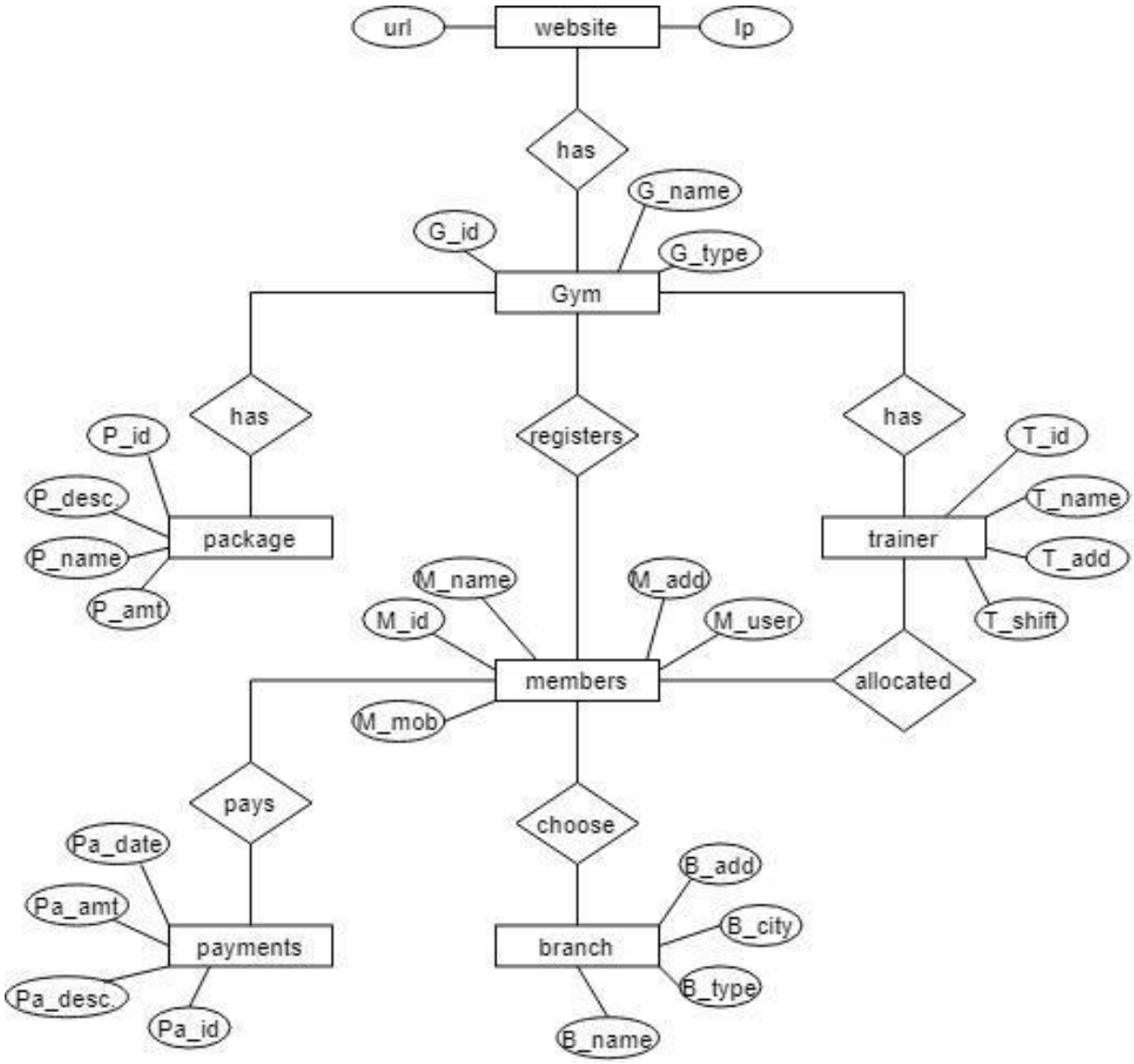
Whether the proposed system causes any trouble or not,etc

o Various costs carried by system

1. Technical experts consulting costs
2. Equipment purches cost
3. Communication equipment cost
4. Operating system software cost
5. Application software cost
6. Documentation preparation cost

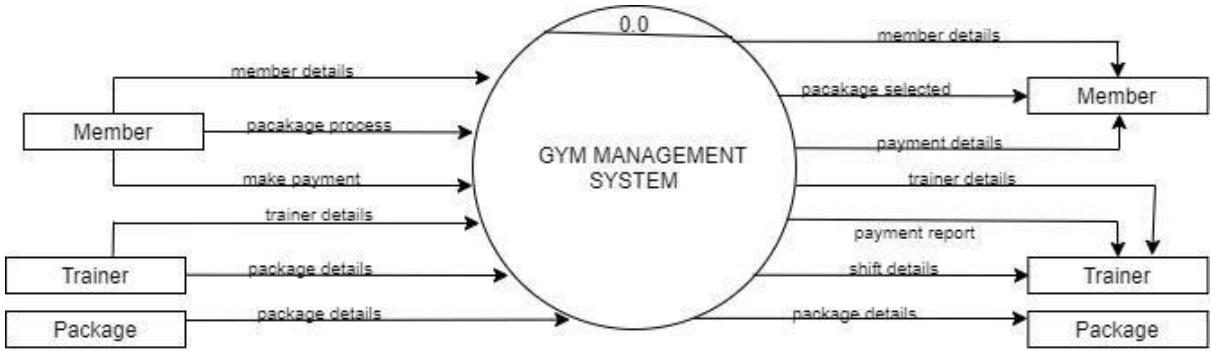
The system delveloped will be user friendly

➢ ER diagram

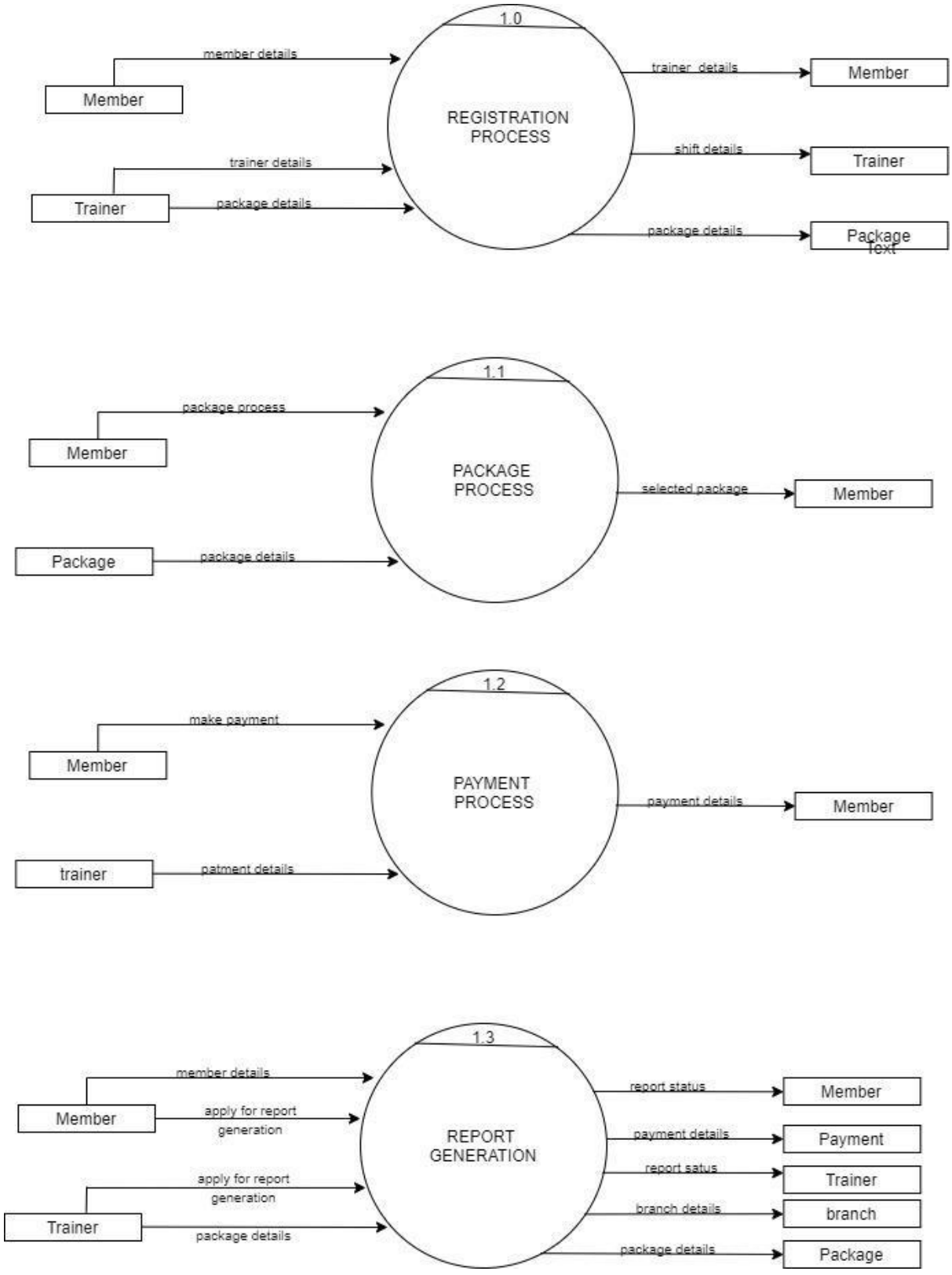

url
website
Ip
has
G_name
G_id
G_type
Gym
has
registers
has
P_id
T_id
P_desc
T_name
package
trainer
P_name
T_add
P_amt
T_shift
M_name
M_add
M_id
M_user
members
allocated
M_mob
pays
choose
Pa_date
B_add
Pa_amt
B_city
payments
branch
Pa_desc
B_type
Pa_id
B_name

➢ Data Flow Diagram (DFD)

❖ 0/context Level DFD:-



❖ 1 Level DFD:-


1.0
REGISTRATION PROCESS
member details
Member
trainer details
Trainer
package details
trainer details
shift details
package details
Member
Trainer
Package
1.1
PACKAGE PROCESS
package process
Member
Package
package details
selected package
Member
1.2
PAYMENT PROCESS
make payment
Member
trainer
patment details
payment details
Member
1.3
REPORT GENERATION
member details
Member
apply for report generation
apply for report generation
Trainer
package details
report status
payment details
report satus
branch details
package details
Member
Payment
Trainer
branch
Package

```
postgres=# \d
```

List of relations

```
 Schema |   Name   | Type  |  Owner
--------+----------+-------+----------
 public | branch   | table | postgres
 public | gym      | table | postgres
 public | member   | table | postgres
 public | package  | table | postgres
 public | pay_memb | table | postgres
 public | payment  | table | postgres
 public | trainer  | table | postgres
(7 rows)
```

- postgres=# select * from gym;

```
 gym_id | gym_name  |       gym_type       |     gym_desc      |
gym_addr
--------+-----------+----------------------+-------------------+-------
---
1001 | ideal fit | crossfit             | yoga included     | pune.mh
1002 | ideal fit | crossfit             | yoga,aro included |
PCMC.mh
1003 | uro fit   | crossfit             | personal.tra inc  | wakad.mh
     (3 rows)
```

- postgres=# select * from trainer;

```
 t_id | t_name |  t_add   | t_shift |  t_email  | gym_id  ------+------
--+---------+---------+----------+--------
1 | ramesh | rahatani | mrng    | ram@gmail |   1001
2 | suresh | pipmri   | eve     | sur@gmail |   1002
3 | mahesh | wakad    | mrng    | mah@gmail |   1003
(3 rows)
```

- postgres=# select * from member;

```
 m_id | m_name  | m_no  | m_addr | m_username | gym_id | t_id  ------+-
--------+-------+--------+------------+--------+------
11 | rushi   | 23456 | pimpri | rushik     |   1001 |    1
12 | omkar   | 23457 | wakad  | omkark     |   1002 |    2
13 | harshal | 23458 | sangvi | harshalk   |   1003 |    3
(3 rows)
```

❖ postgres=# select * from package;

```
 p_id |       p_name        |   p_amt   |  p_type  |       p_desc       |
gym_id  ------+---------------------+----------+----------+-------------
------+--------
 2357 | crossfit            |   $500.00 | monthly  | avail for a month
|   1001
 2355 | crossfit+yoga       | $1,200.00 | 3 months | avail for 3 months
|   1002
 2350 | crossfit+yoga+zumba | $2,200.00 | 6 months | avail for 6 months
|   1003
(3 rows)
```

❖ postgres=# select * from branch;

```
    b_name    |       b_type       | b_city |   b_add    ----
----------+--------------------+--------+----------  Nj.s
fitness | crossfit+yoga      | nagpur | nagpur.mh  golds
| crossfit+yoga(ALL) | nashik | nashik.mh  dynamic      |
crossfit           | mumbai | mumbai.mh
(3 rows)
```

❖ postgres=# select * from payment;

```
 pay_id |  pay_date  |  pay_amt  |   pay_desc    --------+------------
+----------+--------------
      7 | 2018-03-02 |   $800.00 | remain 300
      5 | 2018-12-22 | $3,000.00 | remain 500
      3 | 2018-09-16 | $5,000.00 | payment done
(3 rows)
```

❖ postgres=# select * from pay_memb;

```
 m_id | pay_id | status   ------+--------+---------
11 |      7 | pending
12 |      5 | pending
13 |      3 | done
(3 rows)
```

::NESTED QUERIES::

- postgres=# select * from branch where b_name like 'N%';

```
b_name    |    b_type     | b_city |   b_add    --------------+-
--------------+--------+-----------
 Nj.s fitness | crossfit+yoga | nagpur | nagpur.mh
(1 row)
```

- postgres=# select pay_desc from payment where extract(month from pay_date) =03;

```
  pay_desc   ------
------  remain 300
(1 row)
```

- postgres=# select count(*) from package where p_id>2350;

```
count  -------      2
(1 row)
```

- postgres=# select * from branch order by b_name asc;

```
b_name    |       b_type       | b_city |   b_add    -
-------------+--------------------+--------+----------
-  dynamic      | crossfit           | mumbai |
mumbai.mh  golds        | crossfit+yoga(ALL) | nashik
| nashik.mh  Nj.s fitness | crossfit+yoga      |
nagpur | nagpur.mh (3 rows)
```

::VIEWS::

- create or replace view v1 as select gym.* from gym,trainer,member where gym.gym_id=member.gym_id and trainer.t_id=member.t_id and m_id>11;

```
CREATE VIEW

postgres=# select *from v1;  gym_id | gym_name  |       gym_type
|     gym_desc      | gym_addr
--------+----------+---------------------+------------------+-------
---
      1002  | ideal fit | crossfit             | yoga,aro included |
      PCMC.mh
      1003  | uro fit   | crossfit             | personal.tra inc  |
      wakad.mh (2 rows)
```

- create or replace view v2 as select payment.* from payment,pay_memb,member where payment.pay_id=pay_memb.pay_id and member.m_id=pay_memb.m_id and status='pending';

```
CREATE VIEW

postgres=# select *from v2;
 pay_id |  pay_date  |  pay_amt  |  pay_desc   --------+------------+--
---------+------------
      7 | 2018-03-02 |   $800.00 | remain 300
      5 | 2018-12-22 | $3,000.00 | remain 500
(2 rows)
```

➢

A]

```
postgres=#  select * from v1 where gym_id = 1002;
gym_id | gym_name  |       gym_type       |
gym_desc      |
gym_addr
--------+-----------+----------------------+-------------------+-------
---
   1002 | ideal fit | crossfit             | yoga,aro included |
PCMC.mh
(1 row)
```

B]

```
postgres=#  select * from v1 where gym_id < 1005;

 gym_id | gym_name  |       gym_type       |     gym_desc      |
gym_addr
--------+-----------+----------------------+-------------------+-------
---
      1002  | ideal fit | crossfit             | yoga,aro included |
      PCMC.mh
      1003  | uro fit   | crossfit             | personal.tra inc  |
      wakad.mh (2 rows)
```

C]

```
postgres=#  select * from v2 where pay_id = 7;

 pay_id |  pay_date  | pay_amt |  pay_desc   --------+------------+----
-----+------------
      7 | 2018-03-02 | $800.00 | remain 300
(1 row)
```

::FUNCTION::

- postgres=# CREATE OR REPLACE function gym(a in varchar)

```
RETURNS int as'
DECLARE
total
int;

 BEGIN
 select count(*) into total from gym,trainer,member where
gym.gym_id=trainer.gym_id and member.m_id=trainer.m_id and m_name=r;

 return total;
END;
'LANGUAGE 'plpgsql';
CREATE FUNCTION

postgres=# select gym ('a');
gym  -----
   0
(1 row)
```

- postgres=# create or replace function maximum() returns

```
numeric as'
 DECLARE  maxamt
numeric;

BEGIN
 select max(p_amt) into maxamt from package;

 return maxamt;

END;

'language 'plpgsql';
CREATE FUNCTION

postgres=# select maximum();
maximum  ---------
 2200.00
(1 row)
```

- postgres=# create or replace function membcount(mname in varchar)

```
returns int as'
 declare
total
int;
begin
 select count(*) into total from payment,member,pay_memb where
pay_memb.pay_id=payment.pay_id and pay_memb.m_id=memb.m_id and m_name =
mname;
 return total;
 end

'language 'plpgsql';
CREATE FUNCTION

postgres=# select membcount('mname');
membcount  ------------
          0
(1 row)
```

::CURSORS::

- postgres=# create or replace function branchcity(in name varchar(20))

```
returns varchar as'
 declare
 branchrec branch%rowtype;

 b_cur cursor for select * from branch where b_city = name;
 begin
open b_cur;
loop
  fetch b_cur into branchrec;
exit when not found;
   raise notice ''%'',branchrec.b_name;
  end
loop;
 close b_cur;

 return ''Done'';
 end;'
language 'plpgsql';
CREATE FUNCTION
```

- postgres=# select branchcity('mumbai');

```
NOTICE:  dynamic  branchcity  ------------
 Done
(1 row)
```

- postgres=# create or replace function transfer() returns varchar as' declare pm_cur cursor for select * from pay_memb where pay_id in (1,2); pmrec pay_memb%rowtype;
 begin
 open pm_cur;

 fetch pm_cur into pmrec;

loop
  exit when not found;
   update pay_memb set pid = 1 where pid = 2;
  end
loop;
 close pm_cur;

 return ''Done'';
 end;'
language 'plpgsql';

CREATE FUNCTION postgres=#
select transfer(); transfer

 Done
(1 row)

- postgres=# create or replace function gymrec() returns varchar as'
 declare
 gymrec gym%rowtype;

 g_cur cursor for select * from gym where gym_id between 1000 and 1004;
 begin
 open g_cur;

loop
  fetch g_cur into gymrec;
exit when not found;
   raise notice ''%'',gymrec.gym_desc;
end loop;
 close g_cur;

 return ''Done'';
 end;'
language 'plpgsql';
CREATE FUNCTION

postgres=# select gymrec(); NOTICE:
yoga included
NOTICE: yoga,aro included
NOTICE: personal.tra inc
gymrec --------
 Done
(1 row)

::TRIGGERS::

- postgres-# language 'plpgsql'; CREATE

```
FUNCTION
postgres=# create or replace function deleterec() returns
trigger as'
 begin
 raise notice '' member record is being Deleted '';

 return old;
 end;'
language 'plpgsql';
CREATE FUNCTION
```

```
postgres=# create trigger deleterec before delete on member for each
row execute procedure deleterec();
CREATE TRIGGER
```

```
postgres=# delete from member where m_id=12;
NOTICE:   member record is being Deleted
DELETE 1
```

- postgres=# create or replace function payment_check() returns trigger as'
 begin
 if new.pay_id<2 or new.pay_id >4 then

  raise exception '' Invalid id '';

  return null;
  end
if;
 return new;
 end;'
language 'plpgsql';
CREATE FUNCTION

```
postgres=# insert into payment values(1,'01/08/2017','200','remain
700');
ERROR:   Invalid amout
CONTEXT:  PL/pgSQL function payment_check() line 7 at RAISE
```

- postgres=# create or replace function zero() returns

```
trigger as'
 begin
 if new.p_id = ''0'' then

  raise exception '' Zero Rate not allowed '';
   return
null;  end
if;
 return new;
  end;'
language 'plpgsql';
CREATE FUNCTION
```

```
postgres=# create trigger zero before insert or update on package for
each row execute procedure zero();
CREATE TRIGGER
```

```
postgres=# insert into package values(0,'cf','345','15 days','special
pkg');
ERROR:   Zero Rate not allowed
CONTEXT:  PL/pgSQL function zero() line 7 at RAISE
```