University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Jul 23, 2025

Group Number: 32

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Jamie Ma	37180543	jamiema1	jamiema1@student.ubc.ca
Justin Lee	82486556	jlee2004	jlee2004@student.ubc.ca
Michael Mamic	70634225	s9k6v	michael.mamic78@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

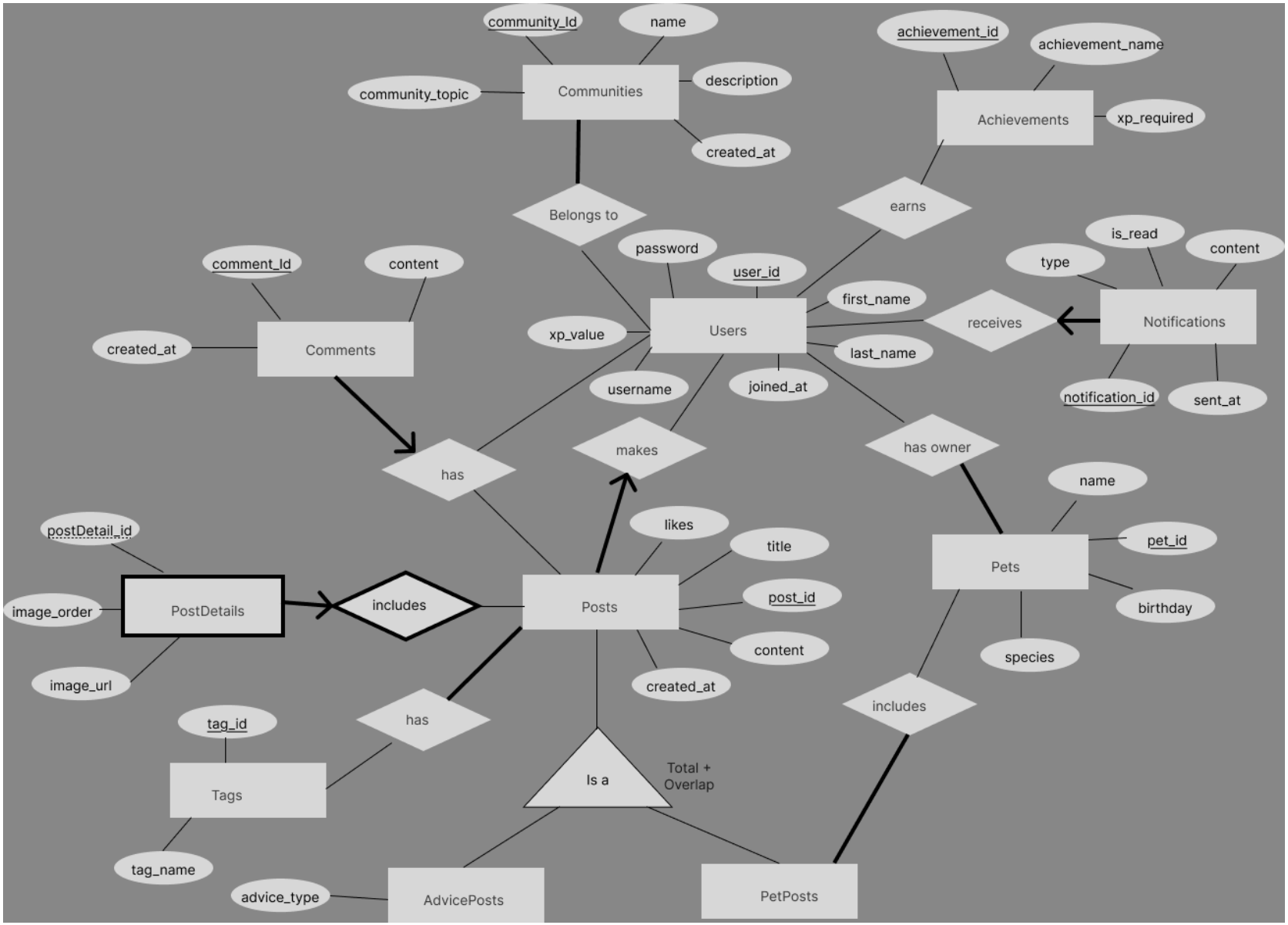
Project Summary - PawLog

PawLog is an application for pet owners to connect and share their experiences and advice about their favourite furry friends. Our application will provide users the ability to interact with a community of like-minded people through the creation of posts about their favourite pet, as well as commenting on and browsing other people's posts.

ER Diagram

Changes:

- added xp_value attribute to Users entity to keep track of xp gained for Achievements
- added achievement_id attribute to Achievments entity to ensure that all keys are integers for consistency
- removed replies relationship to simplify the model
- combined the has and writes relationships into a ternary has relationship to better capture a Comments entity belonging to a Users entity and Posts entity
- added likes attribute to Posts entity to capture the behaviour of users liking a post
- added tag_id attribute to Tags entity to ensure that all keys are integers for consistency


community_Id
name
community_topic
Communities
description
created_at
Belongs to
achievement_id
achievement_name
Achievements
xp_required
earns
password
user_id
first_name
last_name
xp_value
Users
username
joined_at
is_read
content
type
Notifications
receives
notification_id
sent_at
comment_Id
content
created_at
Comments
has
makes
has owner
postDetail_id
image_order
PostDetails
image_url
includes
likes
title
post_id
content
created_at
Posts
name
pet_id
birthday
species
Pets
includes
has
tag_id
Tags
tag_name
Is a
Total +
Overlap
advice_type
AdvicePosts
PetPosts

Schema

Communities(community_id: int, name: string, community_topic: string, description: string, created_at: datetime)
- name is unique and not null, CK
- community_topic is not null
- created_at is not null

BelongsTo(**user_id**, **community_id**)

Users(user_id: int, username: string, password: string, first_name: string, last_name: string, joined_at: date, xp_value: int)
- username is unique and not null, CK
- password is not null
- first_name is not null
- last_name is not null
- joined_at is not null
- xp_value is not null, default 0

Achievements(achievement_id: int, achievement_name: string, xp_required: int)
- achievement_name is unique and not null, CK
- xp_required is not null, default 0

Earns(**user_id**, **achievement_id**)

Notifications(notification_id: int, sent_at: datetime, is_read: boolean, type: string, content: string, **user_id**: int)
- user_id is not null
- sent_at is not null
- is_read is not null, default FALSE
- type is not null

Posts(post_id: int, title: string, content: string, created_at: datetime, likes: int, **user_id**: int)
- user_id is not null
- title is not null
- created_at is not null
- likes is not null, default 0

AdvicePosts(**post_id**: int, advice_type: string)
- advice_type is not null

PetPosts(**post_id**: int)

PetPostsIncludes(**post_id**: int, **pet_id**: int)

PostDetails(postDetail_id: int, image_order: int, image_url: string, **post_id**: int)

- image_order is not null
- image_url is not null

Tags(tag_id: int, tag_name: string)

- tag_name is not null

TaggedWith(**tag_id**: int, **post_id**: int)

Comments(comment_id: int, content: string, created_at: datetime, **user_id**: int, **post_id**: int**)**

- content is not null
- user_id is not null
- post_id is not null
- created_at is not null

Pets(pet_id: int, name: string, birthday: date, species: string, weight: int)

- name is not null
- birthday is not null

HasOwner(**user_id**: int, **pet_id**: int)

Functional Dependencies

Communities:
- community_id → name, community_topic, description, created_at
- name → community_id, community_topic, description, created_at

BelongsTo:
- no FDs, attributes are PKs and FKs

Users:
- user_id → username, password, first_name, last_name, joined_at
- username → first_name, last_name, joined_at, password, user_id

Achievements
- achievement_id → achievement_name, xp_required
- achievement_name → achievement_id, xp_required

Earns:
- no FDs, attributes are PKs and FKs

Notifications
- notification_id → content, type, is_read, sent_at, user_id
- sent_at, user_id → content, type

Posts:
- post_id → title, likes, content, created_at, user_id
- title, created_at, user_id → content

AdvicePosts
- post_id → advice_type

PetPosts(**post_id**: int)
- no FDs, attributes are PKs and FKs

PetPostsIncludes
- no FDs, attributes are PKs and FKs

PostDetails
- postDetail_id, post_id → image_order, image_url
- post_id, image_order → image_url

Tags
- tag_id → tag_name

TaggedWith

- no FDs, attributes are PKs and FKs

Comments

- comment_id → content, created_at, user_id, post_id
- user_id, post_id, created_at → content

Pets

- pet_id → name, species, birthday
- names, species, birthday → weight

HasOwner

- no FDs, all attributes are PKs and FKs

Normalization - BCNF

Communities

- FD1: community_id → name, community_topic, description, created_at
- FD2: name → community_id, community_topic, description, created_at

- { community_id }+ = { community_id, name, community_topic, description, created_at } so it is a super key
- { name }+ = { name, community_id, community_topic, description, created_at } so it is a super key
- since both FD1 and FD2 hold and community_id and name are super keys, the relation is in BCNF

Communities(<u>community_id</u>: int, name: string, community_topic: string, description: string, created_at: datetime)

- name is unique and not null, CK
- community_topic is not null
- created_at is not null

BelongsTo

- no FDs, table is already in BCNF

BelongsTo(**<u>user_id</u>**, **<u>community_id</u>**)

Users

- FD1: user_id → username, password, first_name, last_name, joined_at
- FD2: username → first_name, last_name, joined_at, password, user_id

- { user_id }+ = { user_id, username, password, first_name, last_name, joined_at } so it is a super key
- { username }+ = { username, first_name, last_name, joined_at, password, user_id } so it is a super key
- since both FD1 and FD2 hold and user_id and username are super keys, the relation is in BCNF

Users(<u>user_id</u>: int, username: string, password: string, first_name: string, last_name: string, joined_at: date, xp_value: int)

- username is unique and not null, CK
- password is not null
- first_name is not null
- last_name is not null
- joined_at is not null
- xp_value is not null, default 0

Achievements

- FD1: achievement_id → achievement_name, xp_required
- FD2: achievement_name → achievement_id, xp_required

- { achievement_id }+ = { achievement_id, achievement_name, xp_required } so it is a super key
- { achievement_name }+ = { achievement_name, achievement_id, xp_required } so it is a super key
- since both FD1 and FD2 hold and achievement_id and achievement_name are super keys, the relation is in BCNF

Achievements(<u>achievement_id</u>: string, achievement_name: string, xp_required: int)

- achievement_name is unique and not null, CK
- xp_required is not null, default 0

Earns

- no FDs, attributes are PKs and FKs

Earns(**<u>user_id</u>**, **<u>achievement_id</u>**)

Notifications

- FD1: notification_id → content, type, is_read, sent_at, user_id
- FD2: sent_at, user_id → content, type

- { notification_id }+ = { notification_id, sent_at, user_id, content, type, user_id } so it is a super key
- { sent_at, user_id }+ = { sent_at, user_id, content, type } so it is not a super key and hence not in BCNF
- Since FD2 violates BCNF, we split up Notifications using FD2 into R1(notification_id, sent_at, is_read, user_id) and R2(sent_at, user_id, content, type). FD1 holds in R1 and notification_id is a super key. FD2 holds in R2 and (sent_at, user_id) is a super key. Hence these relations are in BCNF.

Notifications(<u>notification_id</u>: int, sent_at: datetime, is_read: boolean, **user_id**: int)

- (sent_at, user_id) is unique and not null
- is_read is not null, default FALSE

NotificationsContent(**<u>sent_at</u>**: datetime, type: string, content: string, **<u>user_id</u>**: int)

- type is not null

Posts:

- Fd1: post_id → title, likes, content, created_at, user_id
  - { post_id }+ = { title, likes, content, created_at, user_id, post_id } so it is a super key
- Fd2: title, created_at, user_id → content
  - { title, created_at, user_id }+ = { title, content, created_at, user_id } so it is not a super key. Decompose into BCNF: R1(title, likes, created_at, user_id, post_id), R2(title, content, created_at, user_id). Both relations are in BCNF as the key of fd2 is the superkey of R2

Posts(post_id: int, title: string, created_at: datetime, likes: int, **user_id**: int)

- (title, created_at, user_id) is unique and not null
- likes is not null, default 0

PostsContent(**title**: string, content: string, **created_at**: datetime, **user_id**: int)

AdvicePosts

- FD1: post_id → advice_type
- { post_id }+ = { post_id, advice_type }
- since FD1 holds and post_id is a super key, the relation is in BCNF

AdvicePosts(**post_id**: int, advice_type: string)

- advice_type is not null

PetPosts

- no FDs, attributes are PKs and FKs

PetPosts(**post_id**: int)

PetPostsIncludes

- no FDs, attributes are PKs and FKs

PetPostsIncludes(**post_id**: int, **pet_id**: int)

PostDetails(postDetail_id: int, image_order: int, image_url: string, **post_id**: int)

- image_order is not null
- image_url is not null

PostDetails

- Fd1: postDetail_id, post_id → image_order, image_url
  - { postDetail_id, post_id }+ = { postDetail_id, image_order, image_url, post_id} so it is a super key
- Fd2: post_id, image_order → image_url
  - { post_id, image_order }+ = { post_id, image_order, image_url } Since post_id, image_order are not superkeys of PostDetails, it is not in BCNF. Decompose into R1(post_id, image_order, image_url) R2(postDetail_id, image_url, post_id). Both relations are in BCNF as the key of fd2 is the superkey of R1

PostDetails(postDetail_id: int, image_order: int, **post_id**: int)

- image_order is not null

PostDetailsOrder(**image_order**: int, image_url: string, **post_id**: int)

- image_url is not null

Tags
- FD1: tag_id → tag_name
- { tag_id }+ = { tag_id, tag_name }
- since FD1 holds and tag_id is a super key, the relation is in BCNF

Tags(tag_id: int, tag_name: string)
- tag_name is not null

TaggedWith
- no FDs, attributes are PKs and FKs

TaggedWith(**tag_id**: int, **post_id**: int)

Comments(comment_id: int, content: string, created_at: datetime, **user_id**: int, **post_id**: int**)**
- user_id is not null
- post_id is not null
- created_at is not null

Comments
- Fd1: comment_id → comment_id, content, created_at, user_id, post_id
  - { comment_id }+ = { comment_id, content, created_at, user_id, post_id } so it is a super key
- Fd2: user_id, post_id, created_at → content
  - { user_id, post_id, created_at }+ = { content, created_at, user_id, post_id } so it is not a superkey. Therefore, not in BCNF. Decompose into R1(comment_id, created_at, user_id, post_id) R2(user_id, post_id, created_at, content) Both relations are in BCNF as the key of fd2 is the superkey of R2

Comments(comment_id: int, created_at: datetime, **user_id**: int, **post_id**: int**)**
- user_id is not null
- post_id is not null
- created_at is not null

CommentsContent(content: string, **created_at**: datetime, **user_id**: int, **post_id**: int**)**
- content is not null

Pets
- FD1: pet_id → name, species, birthday
- { pet_id }+ = { pet_id, name, species, birthday }
- since FD1 holds and pet_id is a super key, the relation is in BCNF

Pets(pet_id: int, name: string, birthday: date, species: string, weight: int)
- name is not null
- birthday is not null

HasOwner
- no FDs, all attributes are PKs and FKs

HasOwner(**user_id**: int, **pet_id**: int)

DDL

```
CREATE TABLE Communities(
     community_id      INT,
     name              VARCHAR     NOT NULL    UNIQUE,
     community_topic   VARCHAR     NOT NULL,
     description       VARCHAR,
     created_at        DATETIME    NOT NULL,
     PRIMARY KEY (community_id)
);

CREATE TABLE BelongsTo(
     community_id      INT,
     user_id           INT,
     PRIMARY KEY (community_id, user_id),
     FOREIGN KEY (community_id) REFERENCES
Communities(community_id),
     FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE Users(
     user_id           INT,
     username          VARCHAR     NOT NULL   UNIQUE,
     password          VARCHAR     NOT NULL,
     first_name        VARCHAR     NOT NULL,
     last_name         VARCHAR     NOT NULL,
     joined_at         DATE        NOT NULL,
     xp_value          INT         NOT NULL   DEFAULT 0,
     PRIMARY KEY (user_id)
);

CREATE TABLE Achievements(
     achievement_id    INT,
     achievement_name  VARCHAR     NOT NULL    UNIQUE,
     xp_required       INT         NOT NULL    DEFAULT 0,
     PRIMARY KEY (achievement_id)
);

CREATE TABLE Earns(
     achievement_id    INT,
     user_id           INT,
     PRIMARY KEY (achievement_id, user_id),
```

```
     FOREIGN KEY (achievement_id) REFERENCES
Achievements(achievement_id),
     FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE Notifications(
     notification_id   INT,
     sent_at           DATETIME    NOT NULL,
     is_read           BOOLEAN     NOT NULL    DEFAULT FALSE,
     type              VARCHAR     NOT NULL,
     user_id           INT         NOT NULL,
     UNIQUE (sent_at, user_id),
     PRIMARY KEY (notification_id),
     FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE NotificationsContent(
     sent_at           DATETIME,
     type              VARCHAR     NOT NULL,
     content           VARCHAR,
     user_id           INT,
     PRIMARY KEY (sent_at, user_id),
     FOREIGN KEY (sent_at, user_id) REFERENCES
Notifications(sent_at, user_id)
);


CREATE TABLE Posts(
     post_id           INT,
     title             VARCHAR     NOT NULL,
     created_at        DATETIME    NOT NULL,
     likes             INT         NOT NULL    DEFAULT 0,
     user_id           INT         NOT NULL,
     PRIMARY KEY (post_id),
     UNIQUE (title, created_at, user_id),
     FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE PostsContent(
     title             VARCHAR,
     content           VARCHAR,
     created_at        DATETIME,
     user_id           INT,
     PRIMARY KEY (title, created_at, user_id),
     FOREIGN KEY (title, created_at, user_id),
```

```
      REFERENCES Posts(title, created_at, user_id)
);


CREATE TABLE AdvicePosts(
     post_id          INT,
     advice_type      VARCHAR     NOT NULL,
     PRIMARY KEY (post_id),
     FOREIGN KEY (post_id) REFERENCES Posts(post_id)
);

CREATE TABLE PetPosts(
     post_id          INT,
     PRIMARY KEY (post_id),
     FOREIGN KEY (post_id) REFERENCES Posts(post_id)
);

CREATE TABLE PetPostsIncludes(
     post_id          INT,
     pet_id           INT,
     PRIMARY KEY (post_id, pet_id),
     FOREIGN KEY (post_id) REFERENCES Posts(post_id),
     FOREIGN KEY (pet_id) REFERENCES Pets(pet_id)
);

CREATE TABLE PostDetails(
     postDetail_id    INT,
     image_order      INT,
     post_id          INT,
     PRIMARY KEY (postDetail_id, post_id),
     FOREIGN KEY (post_id) REFERENCES Posts(post_id)
);

CREATE TABLE PostDetailsOrder(
     image_order      INT,
     image_url        VARCHAR,
     post_id          INT,
     PRIMARY KEY (image_order, post_id),
     FOREIGN KEY (image_order, post_id) REFERENCES
PostDetails(image_order, post_id)
);

CREATE TABLE Tags(
     tag_id      INT,
     tag_name    VARCHAR NOT NULL,
```

```
     PRIMARY KEY (tag_id)
);

CREATE TABLE TaggedWith(
     tag_id      INT,
     post_id     INT,
     PRIMARY KEY (tag_id, post_id),
     FOREIGN KEY (tag_id) REFERENCES Tags(tag_id),
     FOREIGN KEY (post_id) REFERENCES Posts(post_id)
);

CREATE TABLE Comments(
     comment_id        INT,
     created_at        DATETIME    NOT NULL,
     user_id           INT         NOT NULL,
     post_id           INT         NOT NULL,
     PRIMARY KEY (comment_id),
     UNIQUE (created_at, user_id, post_id),
     FOREIGN KEY (user_id) REFERENCES Users(user_id),
     FOREIGN KEY (post_id) REFERENCES Posts(post_id)
);

CREATE TABLE CommentsContent(
     content           VARCHAR     NOT NULL,
     created_at        DATETIME,
     user_id           INT,
     post_id           INT,
     PRIMARY KEY (created_at, user_id, post_id),
     FOREIGN KEY (created_at, user_id, post_id) REFERENCES
Comments(created_at, user_id, post_id)
);

CREATE TABLE Pets(
     pet_id            INT,
     name              VARCHAR   NOT NULL,
     birthday          DATE      NOT NULL,
     species           VARCHAR,
     PRIMARY KEY (pet_id)
);

CREATE TABLE HasOwner(
     user_id           INT,
     pet_id            INT,
     PRIMARY KEY (user_id, pet_id),
     FOREIGN KEY (user_id) REFERENCES Users(user_id),
```

```
    FOREIGN KEY (pet_id) REFERENCES Pets(pet_id)
);
```

DML

```
INSERT INTO Users (user_id, username, password, first_name,
last_name, joined_at, xp_value)
VALUES
  (1, 'user1', 'pass1234', 'Justin', 'Lee', '2024-06-01', 120),
  (2, 'user2', 'password', 'Min', 'Kim', '2024-06-15', 80),
  (3, 'user3', 'qwerty78', 'Ana', 'Sanchez', '2024-07-01', 200),
  (4, 'user4', 'hello123', 'Dev', 'Patel',   '2024-07-05', 50),
  (5, 'user5', 'mypassword', 'Emily',  'Cho','2024-07-10', 0
);

INSERT INTO Communities (community_id, name, community_topic,
description, created_at)
VALUES
  (1, 'PawPals', 'Dogs', 'A friendly community for dog owners and
lovers.', '2024-06-01 10:00:00'),
  (2, 'MeowWorld', 'Cats', 'Share stories, tips, and pictures of your
feline friends.', '2024-06-05 14:30:00'),
  (3, 'CritterCorner', 'Small Pets', 'Hamsters, rabbits, guinea pigs
— all small pets welcome!', '2024-06-10 09:15:00'),
  (4, 'AquaSphere', 'Aquarium Pets', 'Discuss fish tanks,
aquascaping, and aquatic pets.',  '2024-06-15 18:45:00'),
  (5, 'FeatheredFam',  'Birds', 'For bird owners and enthusiasts of
all feathered friends.', '2024-06-20 12:00:00'
);

INSERT INTO BelongsTo (community_id, user_id)
VALUES
  (1, 1),
  (2, 2),
  (3, 3),
  (4, 4),
  (5, 5);

INSERT INTO Achievements (achievement_id, achievement_name,
xp_required)
VALUES
  (1, 'First Post!',            10),
  (2, 'Helpful Pet Owner',      50),
  (3, 'Pet Photo Pro',          100),
```

```
  (4, 'Community Contributor',   200),
  (5, 'Ultimate Pet Lover',      500);

INSERT INTO Earns (achievement_id, user_id)
VALUES
  (1, 1),
  (2, 2),
  (3, 3),
  (4, 4),
  (5, 5);


INSERT INTO Notifications (notification_id, sent_at, is_read, type,
user_id)
VALUES
  (1, '2024-07-20 09:00:00', FALSE, 'achievement', 1),
  (2, '2024-07-20 09:15:00', FALSE, 'message', 2),
  (3, '2024-07-20 09:30:00', TRUE,  'alert', 3),
  (4, '2024-07-20 09:45:00', FALSE, 'reminder', 4),
  (5, '2024-07-20 10:00:00', FALSE, 'achievement', 5);

INSERT INTO NotificationsContent (sent_at, type, content, user_id)
VALUES
  ('2024-07-20 09:00:00', 'achievement', 'You earned your first
badge!', 1),
  ('2024-07-20 09:15:00', 'message', 'Welcome to the PawPals
community!', 2),
  ('2024-07-20 09:30:00', 'alert', 'Your post received 5 likes.', 3),
  ('2024-07-20 09:45:00', 'reminder', 'Time to feed your aquarium
fish!', 4),
  ('2024-07-20 10:00:00', 'achievement', 'Reached Helpful Pet Owner
level!', 5);


INSERT INTO Posts (post_id, title, created_at, likes, user_id)
VALUES
  (1, 'My First Dog Walk',      '2024-07-01 08:00:00', 10, 1),
  (2, 'Cat Nutrition Tips',     '2024-07-02 09:30:00', 5, 2),
  (3, 'Setting up a Hamster Cage', '2024-07-03 11:00:00', 8, 3),
  (4, 'Best Aquarium Plants',   '2024-07-04 14:45:00', 12, 4),
  (5, 'Bird Training Basics',   '2024-07-05 16:20:00', 7, 5);

INSERT INTO PostsContent (title, content, created_at, user_id)
VALUES
```

```
  ('My First Dog Walk', 'Today I took my dog for a 30-minute walk in
the park.', '2024-07-01 08:00:00', 1),
  ('Cat Nutrition Tips', 'Feeding your cat a balanced diet is crucial
for their health.', '2024-07-02 09:30:00', 2),
  ('Setting up a Hamster Cage', 'Ensure the cage is spacious and
clean for your hamster.', '2024-07-03 11:00:00', 3),
  ('Best Aquarium Plants', 'Some plants help keep your aquarium clean
and oxygenated.', '2024-07-04 14:45:00', 4),
  ('Bird Training Basics', 'Start training your bird with simple
commands and treats.', '2024-07-05 16:20:00', 5);


INSERT INTO AdvicePosts (post_id, advice_type)
VALUES
  (1, 'Dog Training'),
  (2, 'Cat Health'),
  (3, 'Small Pet Care'),
  (4, 'Aquarium Maintenance'),
  (5, 'Bird Behavior');

INSERT INTO PetPosts (post_id)
VALUES
  (1),
  (2),
  (3),
  (4),
  (5);

INSERT INTO Pets (pet_id, name, birthday, species)
VALUES
  (1, 'Buddy',     '2018-03-15', 'Dog'),
  (2, 'Whiskers',  '2020-07-22', 'Cat'),
  (3, 'Nibbles',   '2021-01-10', 'Hamster'),
  (4, 'Goldie',    '2019-11-05', 'Fish'),
  (5, 'Tweety',    '2022-05-30', 'Bird');


INSERT INTO HasOwner (user_id, pet_id)
VALUES
  (1, 1),
  (2, 2),
  (3, 3),
  (4, 4),
  (5, 5);
```

```
INSERT INTO PetPostsIncludes (post_id, pet_id)
VALUES
  (1, 1),
  (2, 2),
  (3, 3),
  (4, 4),
  (5, 5);



INSERT INTO PostDetails (postDetail_id, image_order, post_id)
VALUES
  (1, 1, 1),
  (2, 2, 1),
  (3, 1, 2),
  (4, 1, 3),
  (5, 1, 4);


INSERT INTO PostDetailsOrder (image_order, image_url, post_id)
VALUES
  (1, 'https://example.com/images/dog1.jpg', 1),
  (2, 'https://example.com/images/dog2.jpg', 1),
  (1, 'https://example.com/images/cat1.jpg', 2),
  (1, 'https://example.com/images/hamster1.jpg', 3),
  (1, 'https://example.com/images/fish1.jpg', 4);


INSERT INTO Tags (tag_id, tag_name) VALUES
  (1, 'Dogs'),
  (2, 'Cats'),
  (3, 'Aquarium'),
  (4, 'Birds'),
  (5, 'Hamsters');

INSERT INTO TaggedWith (tag_id, post_id) VALUES
  (1, 1),
  (2, 2),
  (5, 3),
  (3, 4),
  (4, 5);

INSERT INTO Comments (comment_id, created_at, user_id, post_id)
VALUES
```

```
  (1, '2025-07-01 10:15:00', 1, 1),
  (2, '2025-07-01 10:30:00', 2, 2),
  (3, '2025-07-01 11:00:00', 3, 1),
  (4, '2025-07-01 11:15:00', 4, 3),
  (5, '2025-07-01 11:45:00', 5, 4);

INSERT INTO CommentsContent (content, created_at, user_id, post_id)
VALUES
  ('Such a cute dog!', '2025-07-01 10:15:00', 1, 1),
  ('Where did you adopt her?', '2025-07-01 10:30:00', 2, 2),
  ('Looks like my puppy', '2025-07-01 11:00:00', 3, 1),
  ('So fluffy!!', '2025-07-01 11:15:00', 4, 3),
  ('That’s adorable.', '2025-07-01 11:45:00', 5, 4);
```