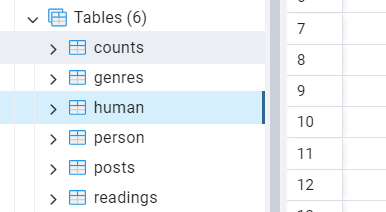
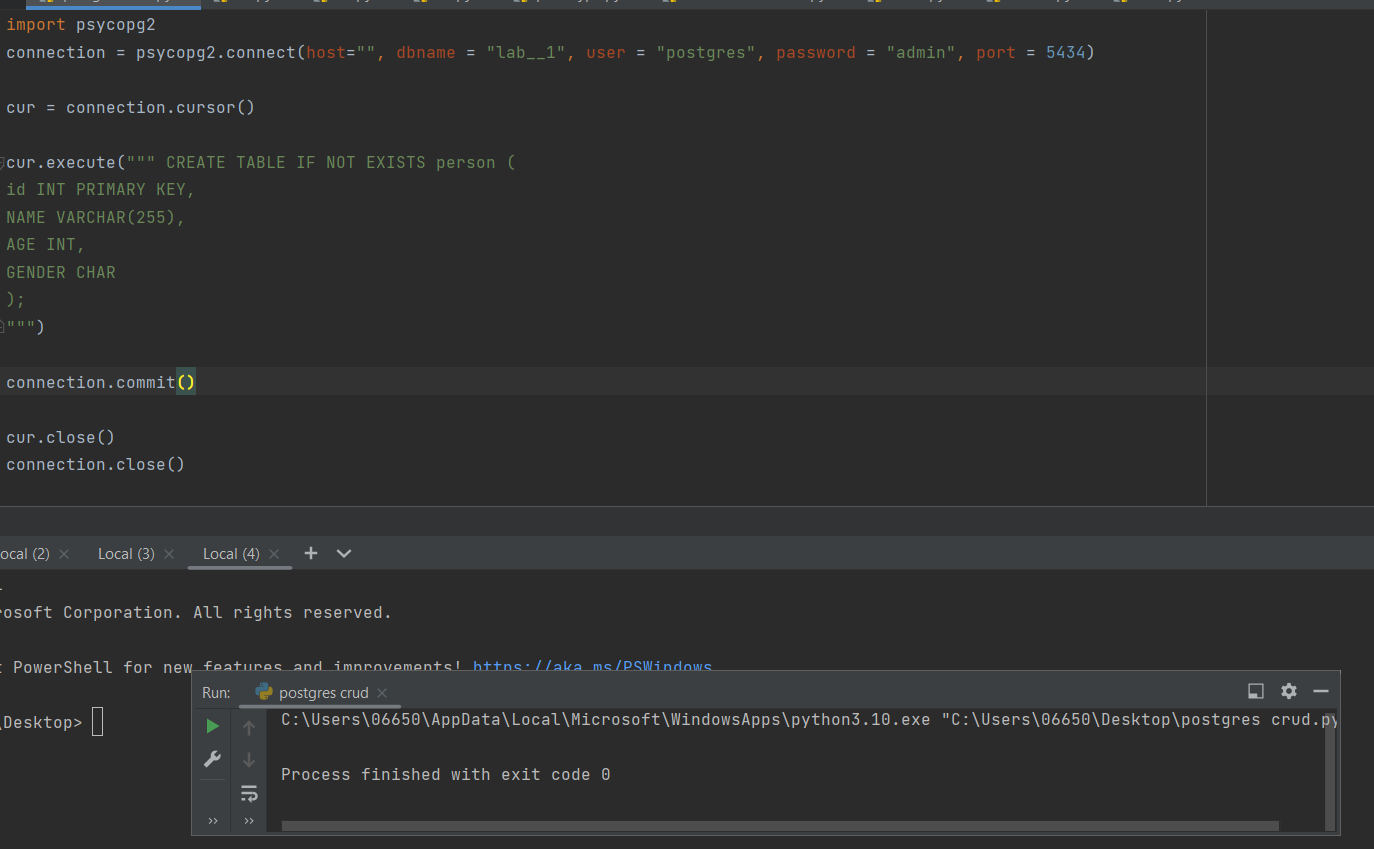
Звіт з бд

лр 3(3,4)

Гніздовський Максим



Процедури, тригери, транзакції, індекси

CREATE OR REPLACE FUNCTION create\_reader(

username varchar,

date\_of\_birth timestamp,

role varchar

)

RETURNS integer

AS $$

BEGIN

INSERT INTO readers (username, date\_of\_birth, role)

VALUES ($1, $2, $3);

RETURN LASTVAL();

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION create\_post(

title varchar,

body text,

genres\_id integer

)

RETURNS integer

AS $$

BEGIN

INSERT INTO posts (title, body, genres\_id)

VALUES ($1, $2, $3);

RETURN LASTVAL();

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION mark\_post\_as\_read(

reader\_id integer,

post\_id integer

)

AS $$

BEGIN

INSERT INTO post\_readings (reader\_id, post\_id)

VALUES ($1, $2);

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trigger\_update\_reader\_last\_login\_on\_login

BEFORE UPDATE ON readers

FOR EACH ROW

WHEN NEW.username = OLD.username

AS $$

BEGIN

UPDATE readers

SET last\_login = CURRENT\_TIMESTAMP

WHERE id = NEW.id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trigger\_increment\_post\_views\_on\_post\_view

AFTER UPDATE ON posts

FOR EACH ROW

WHEN NEW.title <> OLD.title

AS $$

BEGIN

UPDATE posts

SET views = views + 1

WHERE id = NEW.id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trigger\_send\_notification\_on\_new\_post

AFTER INSERT ON posts

FOR EACH ROW

AS $$

BEGIN

-- Send notifications to subscribers of the post's genre

-- (Implementation details depend on the notification system)

END;

$$ LANGUAGE plpgsql;

CREATE INDEX index\_readers\_on\_username ON readers (username);

CREATE OR REPLACE TRIGGER trigger\_update\_reader\_avg\_rating

AFTER UPDATE ON post\_ratings

FOR EACH ROW

WHEN NEW.reader\_id = OLD.reader\_id

AS $$

BEGIN

UPDATE readers

SET avg\_rating = (

SELECT AVG(rating)

FROM post\_ratings

WHERE reader\_id = NEW.reader\_id

)

WHERE id = NEW.reader\_id;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trigger\_log\_post\_update

AFTER UPDATE ON posts

FOR EACH ROW

AS $$

BEGIN

INSERT INTO post\_history (post\_id, old\_title, old\_body, new\_title, new\_body, updated\_at)

VALUES (OLD.id, OLD.title, OLD.body, NEW.title, NEW.body, CURRENT\_TIMESTAMP);

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trigger\_enforce\_unique\_post\_titles

BEFORE INSERT OR UPDATE ON posts

FOR EACH ROW

WHEN (NEW.title = OLD.title OR EXISTS (SELECT 1 FROM posts WHERE title = NEW.title))

AS $$

BEGIN

RAISE EXCEPTION 'Duplicate post title. Please choose a unique title.';

END;

$$ LANGUAGE plpgsql;

CREATE INDEX index\_posts\_on\_title ON posts (title);

CREATE INDEX index\_genres\_on\_name ON genres (name);

CREATE INDEX idx\_title ON posts(title);

CREATE INDEX idx\_human\_id ON posts(human\_id);

CREATE INDEX idx\_genres\_id ON posts(genres\_id);

explain select \* from posts where

ALTER TABLE posts ADD COLUMN updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP;

CREATE OR REPLACE FUNCTION update\_timestamp() RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = NOW();

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update\_posts\_timestamp

BEFORE UPDATE ON posts

FOR EACH ROW

EXECUTE FUNCTION update\_timestamp();

BEGIN;

INSERT INTO posts (id, title, bodyd, human\_id, genres\_id)

VALUES (7, 'New Title', 'New body content', 5, 6);

COMMIT;

BEGIN;

UPDATE posts

SET title = 'Updated Title', bodyd = 'Updated body posts'

WHERE id = 1;

COMMIT;

BEGIN;

DELETE FROM posts

WHERE id = 2;

COMMIT;

CREATE OR REPLACE FUNCTION count\_contents() RETURNS INTEGER AS $$

DECLARE

posts\_count INTEGER;

BEGIN

SELECT COUNT(\*) INTO content\_count FROM posts;

RETURN posts\_count;

END;

$$ LANGUAGE plpgsql;