**Stored Procedures**

**Trigger-1 Session to History**

⇒

CREATE TRIGGER remove\_add\_trigger

AFTER DELETE ON Session

FOR EACH ROW EXECUTE remove\_add();

CREATE OR REPLACE FUNCTION remove\_add() RETURN TRIGGER as $remove\_add\_trigger$

BEGIN

INSERT INTO History(Media\_id, User\_id, Watched\_at, User\_Rating, Left\_at, Screen, Wached\_for) VALUES ( OLD.Media\_id, OLD.User\_id, OLD.watched\_at, NULL, NULL, OLD.Screen, NOW() - OLD.watched\_at);

RETURN NULL;

END

$remove\_add\_trigger$ LANGUAGE pgsql;

⇒

CREATE OR REPLACE FUNCTION hours\_of\_activity(

IN uid INT,

OUT time\_spent TIME(0)

)

AS $BODY$

BEGIN

SELECT INTO time\_spent SUM(Watched\_for)

FROM History WHERE User\_ID = uid;

END; $BODY$

LANGUAGE plpgsql;

⇒

CREATE OR REPLACE FUNCTION avg\_age\_for\_a\_genre(

IN genre VARCHAR(50),

OUT avg FLOAT

)

AS $BODY$

BEGIN

SELECT AVG(Age) as Age

FROM users JOIN HISTORY ON users.id = History.user\_id

JOIN Media ON History.media\_id = Media.id

WHERE Genre\_name = genre;

END; $BODY$

LANGUAGE plpgsql;

⇒

CREATE OR REPLACE FUNCTION topfive () RETURNS SETOF Media AS $BODY$

DECLARE e Users%rowtype;

BEGIN

FOR e IN SELECT IMDb FROM Media ORDER BY IMDb DESC LIMIT 5

LOOP

RETURN NEXT e;

END LOOP;

RETURN;

END $BODY$ LANGUAGE plpgsql;

⇒

CREATE OR REPLACE FUNCTION region\_pref(IN genre VARCHAR(15)) RETURNS SETOF Media AS $BODY$

DECLARE e Media%rowtype;

BEGIN

FOR e IN SELECT Genre.Name, User.Location, Count(\*) as count FROM History NATURAL JOIN Users NATURAL JOIN Media NATURAL JOIN Genre GROUP BY Location ORDER BY count DESC;

LOOP

RETURN NEXT e;

END LOOP;

RETURN;

END $BODY$ LANGUAGE ‘plpgsql’;