ASSIGNMENT 1: PROOF OF CREATION FUNCTION, TRIGGER AND SEQUENCE

NAME: NURUL BALQIS BINTI MOKHTAR

STUDENT NUMBER: 48365363

COURSE: INFS2200

```
a1=#
a1=# ALTER TABLE Events
a1-# ADD CONSTRAINT CK_EVENT_TYPE
a1-# CHECK(event_type IN('Loan', 'Return', 'Hold', 'Loss'));
ALTER TABLE
a1=#
```

Figure A: Q1.1

```
a1=# --create function
a1=# CREATE OR REPLACE FUNCTION UDF_BI_GUARDIAN() RETURNS TRIGGER AS $$
a1$# BEGIN
a1$# --check age > 18
a1$# IF(NEW.dob > CURRENT_DATE - INTERVAL '18 years') THEN
a1$# --check if guardian exist
a1$# IF(NEW.guardian IS NULL OR NOT EXISTS(SELECT 1 FROM Patrons WHERE patron_id = NEW.guardian)) THEN
a1$# RAISE EXCEPTION 'Patrons under 18 years old must have a guardian.';
a1$# END IF;
a1$# END IF;
a1$# RETURN NEW;
a1$# END;
a1$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
a1=# --create trigger
a1=# CREATE TRIGGER BI_GUARDIAN
a1-# BEFORE INSERT ON Patrons
a1-# FOR EACH ROW
a1-# EXECUTE FUNCTION UDF_BI_GUARDIAN();
CREATE TRIGGER
a1=#
```

Figure B: Q2.1

```
a1=#
a1=# --create function
a1=# CREATE OR REPLACE FUNCTION UDF_BI_EMAIL_ADDR() RETURNS TRIGGER AS $$
a1$# BEGIN
a1$# --check patron's age > 18
a1$# IF (NEW.dob <= CURRENT_DATE - INTERVAL '18 years') THEN
a1$# --make sure email provided
a1$# IF(NEW.email_address IS NULL OR NEW.email_address = '')THEN
a1$# RAISE EXCEPTION 'Adults patrons must have email address.';
a1$# END IF;
a1$# END IF;
a1$# RETURN NEW;
a1$# END;
a1$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
a1=# --create trigger
a1=# CREATE TRIGGER BI_EMAIL_ADDR
a1-# BEFORE INSERT ON Patrons
a1-# FOR EACH ROW
a1-# EXECUTE FUNCTION UDF_BI_EMAIL_ADDR();
CREATE TRIGGER
a1=#
```

Figure C: Q2.2

```
a1=# CREATE SEQUENCE ITEM_ID_SEQ
a1-# MINVALUE 1000000000
a1-# MAXVALUE 9999999999
a1-# INCREMENT BY 1
a1-# START WITH 1000000000
a1-# ;
CREATE SEQUENCE
a1=#
```

Figure D: Q3.1

```
a1=#
a1=# --create UDF
a1=# CREATE OR REPLACE FUNCTION UDF_BI_ITEM_ID() RETURNS TRIGGER AS $$
a1$# DECLARE
a1$# seq_number bigint;
a1$# barcode_text varchar;
a1$# checksum_digit int;
a1$# i int;
a1$# sum_digit int := 0;
a1$# BEGIN
a1$# --get next value from sequence
a1$#
a1$# seq_number := nextval('ITEM_ID_SEQ');
a1$# --calculate sum of all digit in seq_number % 10
a1$# FOR i IN 1..10 LOOP
a1$# sum_digit := sum_digit + (substring(seq_number::text, i, 1)::int);
a1$# END LOOP;
a1$# checksum_digit := sum_digit % 10;
a1$# --barcode text 'UQ' prefix, seq_number, checksum digit
a1$# barcode_text := 'UQ' || seq_number || checksum_digit;
a1$# --assign barcode text to PK of new record
a1$# NEW.item_id := barcode_text;
a1$# RETURN NEW;
a1$# END;
a1$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
a1=# --create trigger
a1=# CREATE TRIGGER BI_ITEM_ID
a1-# BEFORE INSERT ON Items
a1-# FOR EACH ROW
a1-# EXECUTE FUNCTION UDF_BI_ITEM_ID();
CREATE TRIGGER
a1=#
```

Figure E: Q3.2

```
a1=# --sequence identification
a1=# SELECT
a1-# c.relname AS sequence_name
a1-# FROM
a1-# pg_class c
a1-# JOIN
a1-# pg_namespace n ON n.oid = c.relnamespace
a1-# WHERE
a1-# c.relkind = 'S'
a1-# AND n.nspname = 'public'
a1-# AND(
a1(# c.relname LIKE 'patrons_%_seq' OR
a1(# c.relname LIKE 'events_%_seq'
a1(# );
     sequence_name
-----------------------
 patrons_patron_id_seq
 events_event_id_seq
(2 rows)

a1=#
```

Figure F: Q3.3

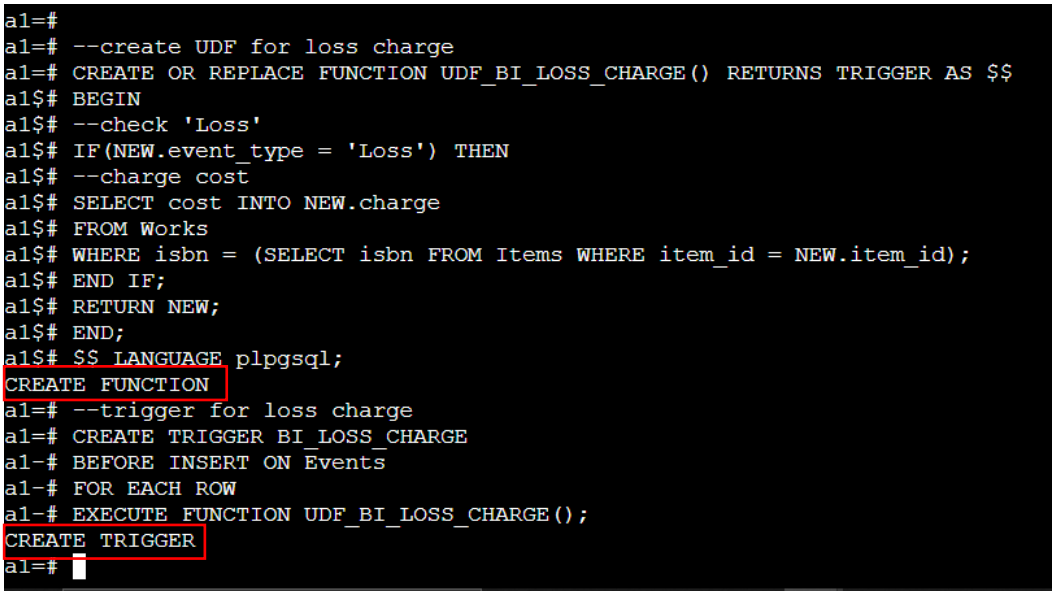
a1=#
a1=# --create UDF for loss charge
a1=# CREATE OR REPLACE FUNCTION UDF_BI_LOSS_CHARGE() RETURNS TRIGGER AS $$
a1$# BEGIN
a1$# --check 'Loss'
a1$# IF(NEW.event_type = 'Loss') THEN
a1$# --charge cost
a1$# SELECT cost INTO NEW.charge
a1$# FROM Works
a1$# WHERE isbn = (SELECT isbn FROM Items WHERE item_id = NEW.item_id);
a1$# END IF;
a1$# RETURN NEW;
a1$# END;
a1$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
a1=# --trigger for loss charge
a1=# CREATE TRIGGER BI_LOSS_CHARGE
a1-# BEFORE INSERT ON Events
a1-# FOR EACH ROW
a1-# EXECUTE FUNCTION UDF_BI_LOSS_CHARGE();
CREATE TRIGGER
a1=#

Figure G: Q4.1

```
a1=# --create function fro missing return
a1=# CREATE OR REPLACE FUNCTION public.udf_ai_missing_return()
a1-#  RETURNS trigger
a1-#  LANGUAGE plpgsql
a1-# AS $function$
a1$# DECLARE
a1$#
a1$# last_loan RECORD;
a1$# BEGIN
a1$#
a1$# --check even = 'Loan'
a1$#
a1$#
a1$# IF(NEW.event_type = 'Loan') THEN
a1$#
a1$# --find the same item id, the most recent 'Loan'
a1$#
a1$# SELECT * INTO last_loan
a1$#
a1$# FROM Events e1
a1$#
a1$# WHERE e1.item_id = NEW.item_id
a1$#
a1$# AND e1.event_type = 'Loan'
a1$#
a1$#
a1$# AND e1.time_stamp <> NEW.time_stamp
a1$#
a1$# ORDER BY e1.time_stamp DESC
a1$#
a1$# LIMIT 1;
a1$#
a1$# --if loan is found, and different borrower, return previous borrower, 1 hr earlier
a1$#
a1$# IF FOUND AND last_loan.patron_id <> NEW.patron_id THEN
a1$#
a1$# -- Insert a 'Return' event 1 hour before the new 'Loan'
a1$#
a1$# PERFORM pg_notify('insert_return', 'Skipping trigger execution');
a1$#
a1$# INSERT INTO Events (patron_id, item_id, event_type, time_stamp, charge)
a1$#
a1$# VALUES (last_loan.patron_id, last_loan.item_id, 'Return', NEW.time_stamp - INTERVAL '1 hour', NULL);
a1$#
a1$# END IF;
a1$#
a1$# END IF;
a1$#
a1$# RETURN NEW;
a1$# END;
a1$# $function$;
CREATE FUNCTION
a1=# --trigger for missing return
a1=# CREATE TRIGGER AI_MISSING_RETURN
a1-# AFTER INSERT ON Events
a1-# FOR EACH ROW
a1-# WHEN (NEW.event_type = 'Loan')
a1-# EXECUTE FUNCTION public.udf_ai_missing_return();
CREATE TRIGGER
a1-#
```

Figure H: Q4.2

```
a1=# CREATE OR REPLACE FUNCTION public.udf_bi_holds()
a1-#  RETURNS trigger
a1-#  LANGUAGE plpgsql
a1-# AS $function$
a1$# DECLARE
a1$# LAST_LOAN_TIME TIMESTAMP;
a1$# BEGIN
a1$#
a1$# --check if item is loss
a1$#
a1$# IF EXISTS(
a1$#
a1$# SELECT 1 FROM EVENTS
a1$#
a1$# WHERE ITEM_ID = NEW.ITEM_ID AND EVENT_TYPE = 'Loss'
a1$#
a1$# )THEN
a1$#
a1$# RAISE EXCEPTION 'Item is loss, cannot be hold.';
a1$#
a1$# END IF;
a1$#
a1$# --check if item is on hold, not returned
a1$#
a1$# IF EXISTS(
a1$#
a1$# SELECT 1
a1$#
a1$# FROM EVENTS
a1$#
a1$# WHERE ITEM_ID = NEW.ITEM_ID
a1$#
a1$# AND EVENT_TYPE = 'Hold'
a1$#
a1$# AND NOT EXISTS(
a1$#
a1$# SELECT 1
a1$#
a1$# FROM EVENTS e2
a1$#
a1$# WHERE e2.ITEM_ID = NEW.ITEM_ID
a1$#
a1$# AND e2.EVENT_TYPE = 'Return'
a1$#
a1$# AND e2.time_stamp > EVENTS.time_stamp
a1$#
```

Figure I: Q4.3

```
a1$#
a1$# )
a1$#
a1$# )THEN
a1$#
a1$# --if return event after hold
a1$#
a1$# RETURN NEW;
a1$#
a1$# --check if it's on loan by another patron
a1$#
a1$# ELSIF EXISTS (
a1$#
a1$# SELECT 1
a1$#
a1$# FROM EVENTS
a1$#
a1$# WHERE ITEM_ID = NEW.ITEM_ID
a1$#
a1$# AND EVENT_TYPE = 'Loan'
a1$#
a1$# AND PATRON_ID <> NEW.PATRON_ID
a1$#
a1$# ) THEN
a1$#
a1$# --expiry of hold 42 days after loan
a1$#
a1$#
a1$# SELECT MAX(time_stamp) INTO LAST_LOAN_TIME
a1$#
a1$# FROM EVENTS
a1$#
a1$# WHERE ITEM_ID = NEW.ITEM_ID
a1$#
a1$# AND EVENT_TYPE = 'Loan';
a1$#
a1$# NEW.TIME_STAMP := LAST_LOAN_TIME + INTERVAL '42 days';
a1$#
a1$# --set hold time to 14 days if none
a1$#  ELSIF EXISTS (
a1$#
a1$# SELECT 1
a1$#
a1$# FROM EVENTS e1
a1$#
a1$# WHERE e1.ITEM_ID = NEW.ITEM_ID
```

Figure J: Q4.3

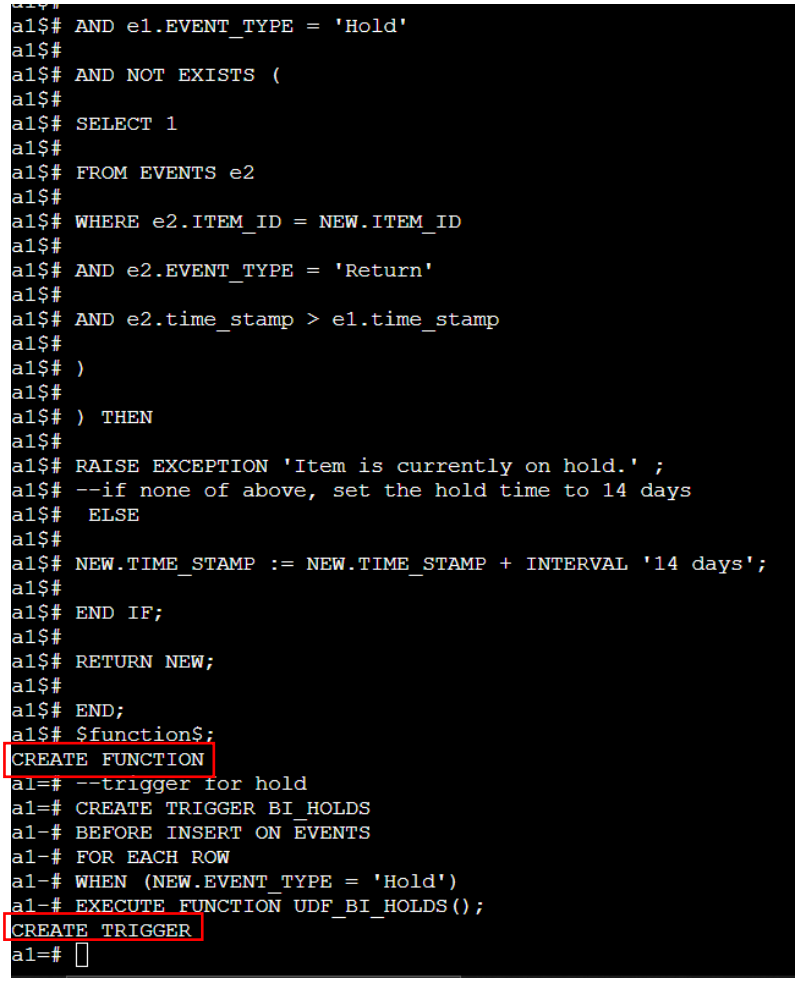
a1$# AND e1.EVENT_TYPE = 'Hold'
a1$#
a1$# AND NOT EXISTS (
a1$#
a1$# SELECT 1
a1$#
a1$# FROM EVENTS e2
a1$#
a1$# WHERE e2.ITEM_ID = NEW.ITEM_ID
a1$#
a1$# AND e2.EVENT_TYPE = 'Return'
a1$#
a1$# AND e2.time_stamp > e1.time_stamp
a1$#
a1$#)
a1$#
a1$#) THEN
a1$#
a1$# RAISE EXCEPTION 'Item is currently on hold.' ;
a1$# --if none of above, set the hold time to 14 days
a1$# ELSE
a1$#
a1$# NEW.TIME_STAMP := NEW.TIME_STAMP + INTERVAL '14 days';
a1$#
a1$# END IF;
a1$#
a1$# RETURN NEW;
a1$#
a1$# END;
a1$# $function$;
CREATE FUNCTION
a1=# --trigger for hold
a1=# CREATE TRIGGER BI_HOLDS
a1-# BEFORE INSERT ON EVENTS
a1-# FOR EACH ROW
a1-# WHEN (NEW.EVENT_TYPE = 'Hold')
a1-# EXECUTE FUNCTION UDF_BI_HOLDS();
CREATE TRIGGER
a1=#

Figure K: Q4.3