Creating Carnival Reports

As Carnival grows, we have been asked to help solve two issues:

1. It has become more and more difficult for the accounting department to keep track of the all the records in the sales table and how much money came in from each sale.

2. HR currently has an overflowing filing cabinet with files on each employee. There's additional files for each dealership. Sorting through all these files when new employees join Carnival and current employees leave is a process that needs to be streamlined. All employees that start at Carnival are required to work shifts at at least two dealerships.

Goals

- Using CREATE to add new tables
- Using triggers
- Using stored procedures
- Using transactions

Practice

Provide a way for the accounting team to track all financial transactions by creating a new table called Accounts Receivable. The table should have the following columns: credit_amount, debit_amount, date_received as well as a PK and a FK to associate a sale with each transaction.

```
create table accountsreceivable (
     accountsreceivable_id SERIAL primary key,
     credit_amount int,
     debit_amount int,
     date_received date,
     sale_id int references sales (sale_id)
)
```

1. Set up a trigger on the Sales table. When a new row is added, add a new record to the Accounts Receivable table with the deposit as credit_amount, the timestamp as date_received and the appropriate sale_id.

```
CREATE or replace FUNCTION add_accountsreceivable()

 RETURNS TRIGGER
 LANGUAGE PlPGSQL
AS $$
BEGIN
 -- trigger function logic
 insert into accountsreceivable (credit_amount,
date_received, sale_id)
 values (NEW.deposit, NEW.purchase_date, NEW.sale_id);
  RETURN NEW;
END;
$$
create or replace TRIGGER new_sale_made
 AFTER INSERT
 ON sales
 FOR EACH ROW
 EXECUTE PROCEDURE add_accountsreceivable();
```

2. Set up a trigger on the Sales table for when the sale_returned flag is updated. Add a new row to the Accounts Receivable table with the deposit as debit_amount, the timestamp as date_received, etc.

```
CREATE or replace FUNCTION
add_returned_accountsreceivable()

 RETURNS TRIGGER
 LANGUAGE PlPGSQL
AS $$
BEGIN
```

```
 -- trigger function logic
      if new.sale_returned = true then
        insert into accountsreceivable
(debit_amount, date_received, sale_id)
        values (NEW.deposit, NOW(), NEW.sale_id);
      end if;
 RETURN NEW;
END;
$$
create or replace TRIGGER new_return_made
 AFTER UPDATE
 ON sales
 FOR EACH ROW
 EXECUTE PROCEDURE
add_returned_accountsreceivable();
```

Practice

Help out HR fast track turnover by providing the following:

1. Create a stored procedure with a transaction to handle hiring a new employee. Add a new record for the employee in the Employees table and add a record to the Dealershipemployees table for the two dealerships the new employee will start at.

```
CREATE OR REPLACE PROCEDURE hireNewEmployee(IN dealershipId1 INT, IN
dealershipId2 INT)
LANGUAGE plpgsql
AS $$
DECLARE
      EmployeeId integer;
BEGIN
      --Add a new record for the employee in the Employees table and
INSERT INTO employees(first_name, last_name, email_address, phone,
employee_type_id)
VALUES ('bob', 'tree', 'bobp@aol.com', '555-444-5555', '3')
RETURNING employee_id INTO EmployeeId;
COMMIT;
      --add a record to the Dealershipemployees table for the two dealerships
the new employee will start at
```

```
INSERT INTO dealershipemployees (dealership_id, employee_id)
VALUES (dealershipId1, EmployeeId);
COMMIT;
INSERT INTO dealershipemployees (dealership_id, employee_id)
VALUES (dealershipId2, EmployeeId);
COMMIT;
END;
$$;
```

2. Create a stored procedure with a transaction to handle an employee leaving. The employee record is removed and all records associating the employee with dealerships must also be removed.

```
CREATE OR REPLACE PROCEDURE dismissedEmployee(IN dismissed_id INT)
LANGUAGE plpgsql
AS $$
BEGIN

	DELETE FROM dealershipemployees
	WHERE employee_id = dismissed_id;

	DELETE FROM employees
	WHERE employee_id = dismissed_id;

END;
$$;

CALL dismissedEmployee(2)
```