Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```
create or replace trigger prevent-parent- delete
before delete ~~on~~ parent- table
for each row
declare
    v- child- cound number;
begin
    select count(*)
    into v- child-count
    from child-table
    where parent-id =:' old.parent- primary - key;
    if v- child- count >0 then
        raise- application error (-20001, 'can not delete parent ~~row~~:
        child records exist.');
        end if;
    end;
    /
```

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
create or replace trigger check_duplicate_value
before inst or update on your_table
for each row
declare
    v_count number;
begin
    select count(*)
    into v_count
    from your_table
    where some_unique_colum = :new.some_unique_column;
    if v_count>0 then
        raise_application_error(-20002, 'duplicate value: this value
        already exists.');
        end if
    end;
    /
```

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```
create or replace trigger check_total_threshold
after insert on your_table
declare
        v_total number;
        v_threshold number := 100000;
begin
      select sum(some_column)
      into v_total
      from your_table;
      if v_total > v_threshold then
           raise_application_error( -20003, 'insertion failed: total
           exceeds threshold.');
             end if;
      end;
      /
```

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
create table column_audit_log (
    log_id number generated as identity,
    table_audited varchar2(30),
    colum_audited varchar2(30),
    row_pk varchar2(100),
    old_value varchar2(1000),
    new_value varchar2(1000),
    changed_by varchar2(30),
    change_date timestamp);
create or replace trigger log_column_changes
after update on your_table
for each row
begin
    if :old.salary != :new.salary then
    insert into colum_audit_log(table_audited, column_audited, row_pk,
    old_value, new_value, changed_by, change_date) values(
    'your_table', 'salary', :old.primary_key, :old.salary,
    :new.salary, user, systemtimestamp);
    end if
    if :old.job_id != :new.jobid then
        insert into column_audit_log(table, column_audited, row_pk,
        old_value, new_value, changed_by, change_date) values(
        'your_table', 'job_id', :old.primary_key, :old.job_id,
        :new.job_id, user, systimestamp);
        end if;
    end;
    /
```

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```
create table activity audit_log(
      log-id number generated as identity,
      table audited varchar2(30),
      dml-action varchar2(10),
      row_pk varchar2(100),
      action-by varchar2(30),
      action-date timestamp);
create or replace trigger log-user-activity
after insert or update or delete on your_table
for each row
declare
      v-action varchar2(10);
begin if inserting then
          v-action := 'insert';
          insert into activity-audit-log(table-audited, dml-action,
          row-pk, action-by, action-date) values('your-table',
          v-action, :new. primary-key, user, systimestamp);
     else if updating then
          v-action := 'update';
          insert into activity-audit-log( table-audited, dml-action
          row-pk, action-by, action-date) values('your-table',
          v-action, :old. primary-key, user, systimestamp);
     elsif deleting then
          v- action := 'delete';
          insert into activity-audit-log(table-audited, dml-action,
          row-pk, action-by, action-date) values('your-table',
          v-action, :old. primary-key, user, systimestamp);
          end if;
     end;
     /
```

Program 7

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
create or replace trigger update_running_total
after insert on orders
for each row
begin
    update sales_summary
   set total_sales = total_sales + :new.amount
   where summary_id = 1;
end;
/
```

Program 8

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
create or replace trigger validate-stock-level
before insert on order-items
for each row
declare
    v-stock number;
begin
   select stock-level
   into v-stock
   from products
   where product-id =: new.product_id;
   if v-stock < :new.quantity then
       raise-application-error(-20004, 'cannot place order:
       insufficient stock for product' || :new.product-id);
   else
      update products
      set stock_level = stock_level-:new.quantity
      where product-id = :new.product_id;
   end if;
end;
/
```