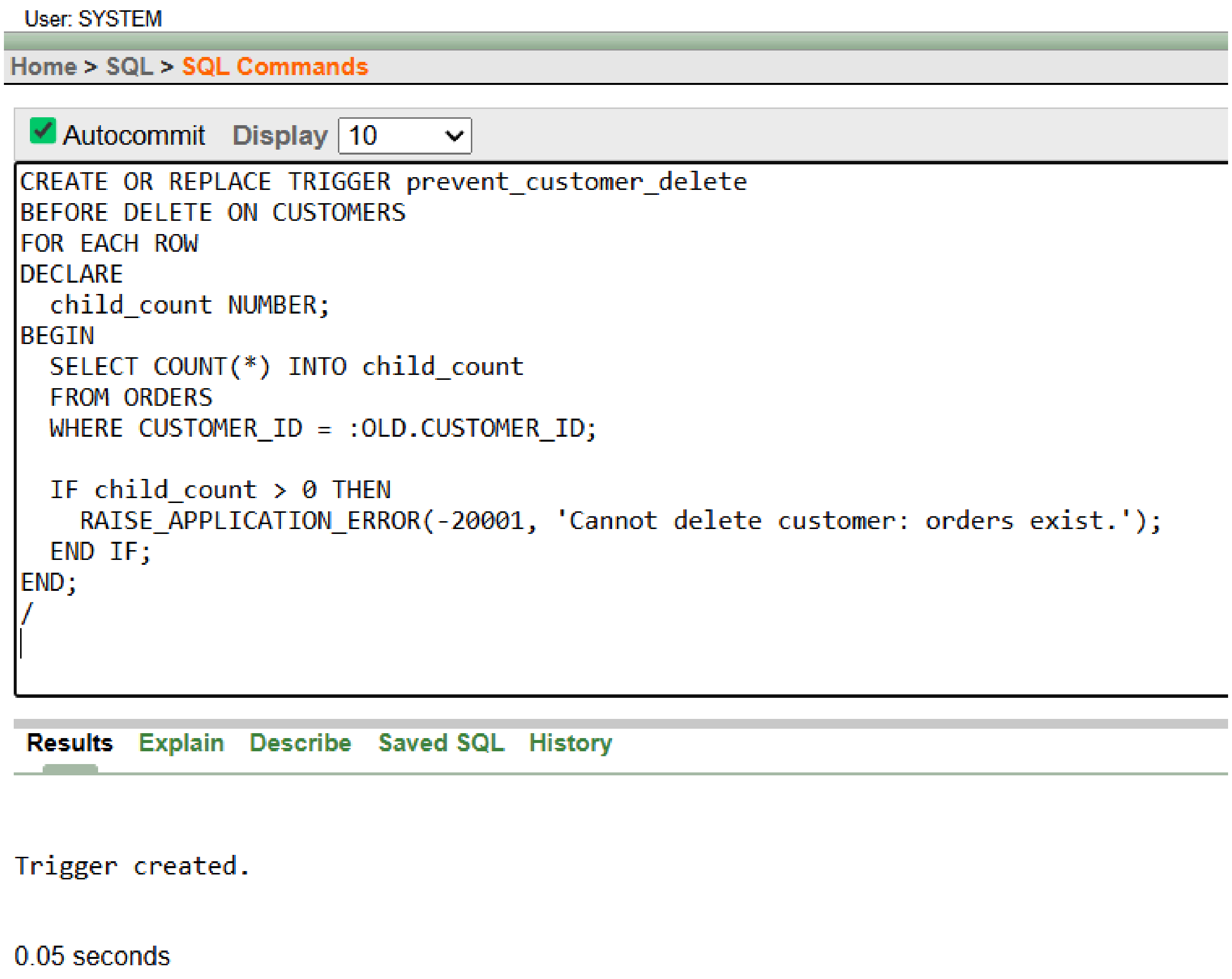
Tab 1



Display: 10

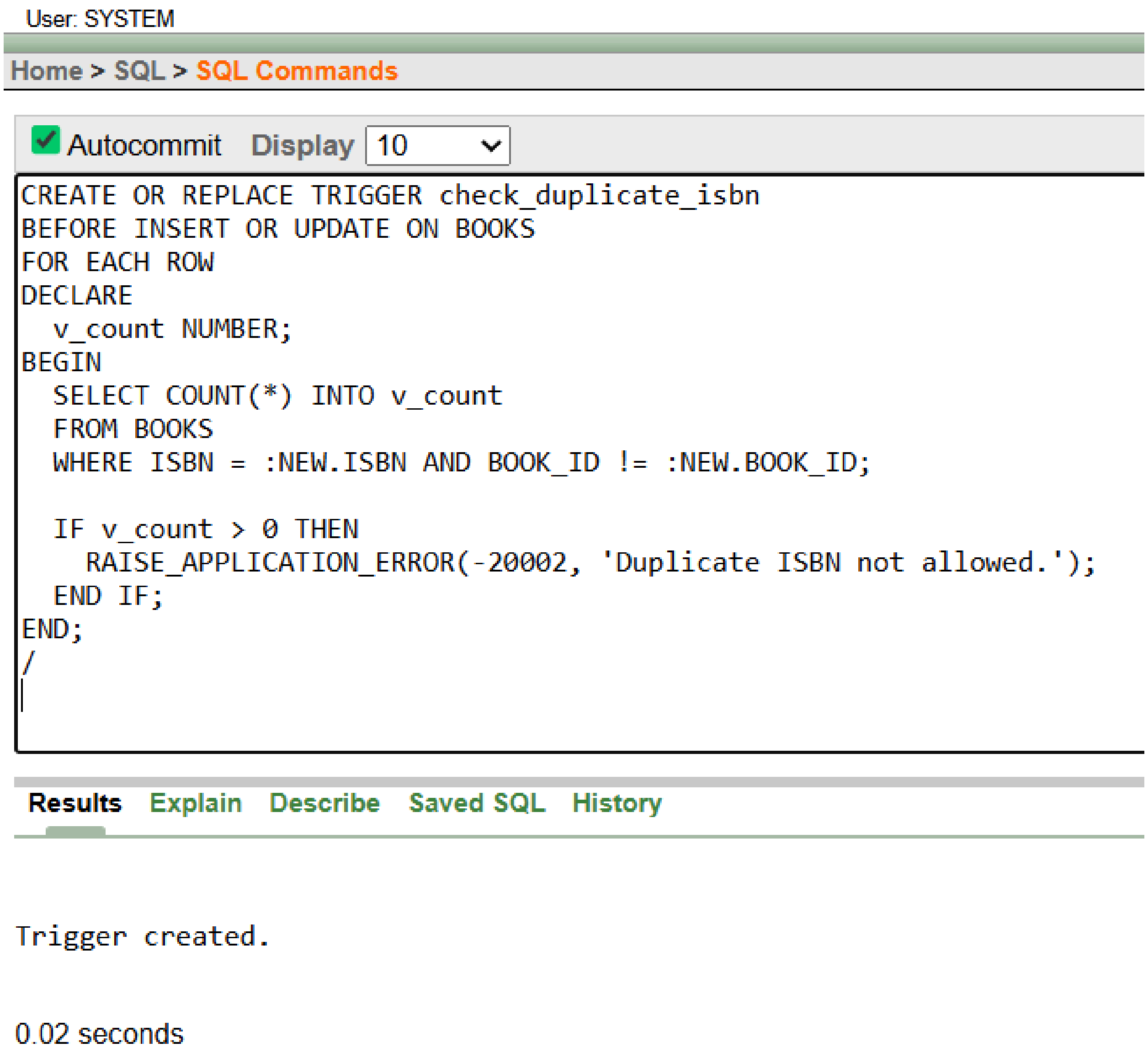
```
CREATE OR REPLACE TRIGGER prevent_customer_delete
BEFORE DELETE ON CUSTOMERS
FOR EACH ROW
DECLARE
  child_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO child_count
  FROM ORDERS
  WHERE CUSTOMER_ID = :OLD.CUSTOMER_ID;

  IF child_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Cannot delete customer: orders exist.');
  END IF;
END;
/
```

```
Trigger created.
```

0.05 seconds

Tab 2



Autocommit [✓] Display 10

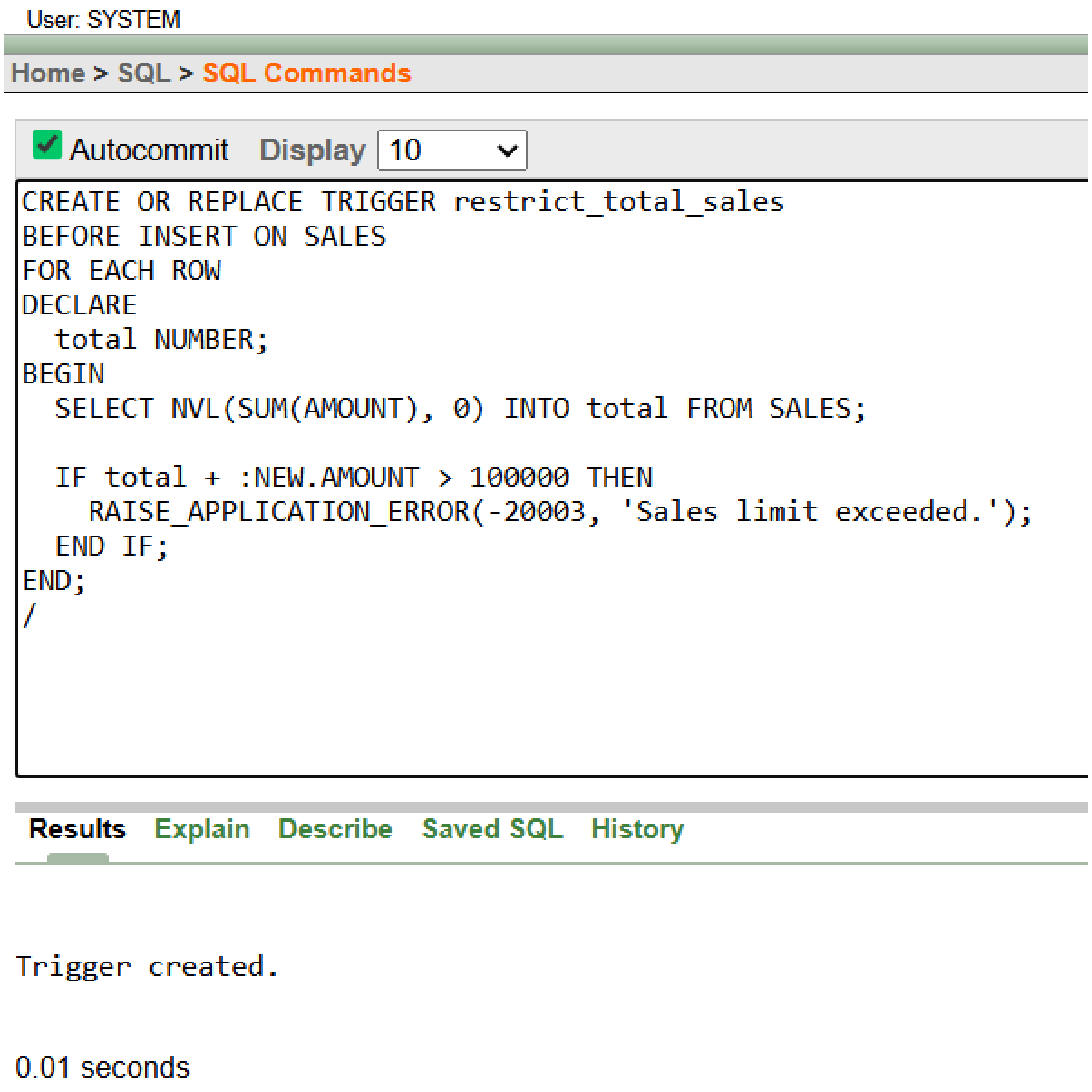
```
CREATE OR REPLACE TRIGGER check_duplicate_isbn
BEFORE INSERT OR UPDATE ON BOOKS
FOR EACH ROW
DECLARE
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count
  FROM BOOKS
  WHERE ISBN = :NEW.ISBN AND BOOK_ID != :NEW.BOOK_ID;

  IF v_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20002, 'Duplicate ISBN not allowed.');
  END IF;
END;
/
```

```
Trigger created.
```

0.02 seconds

Tab 3



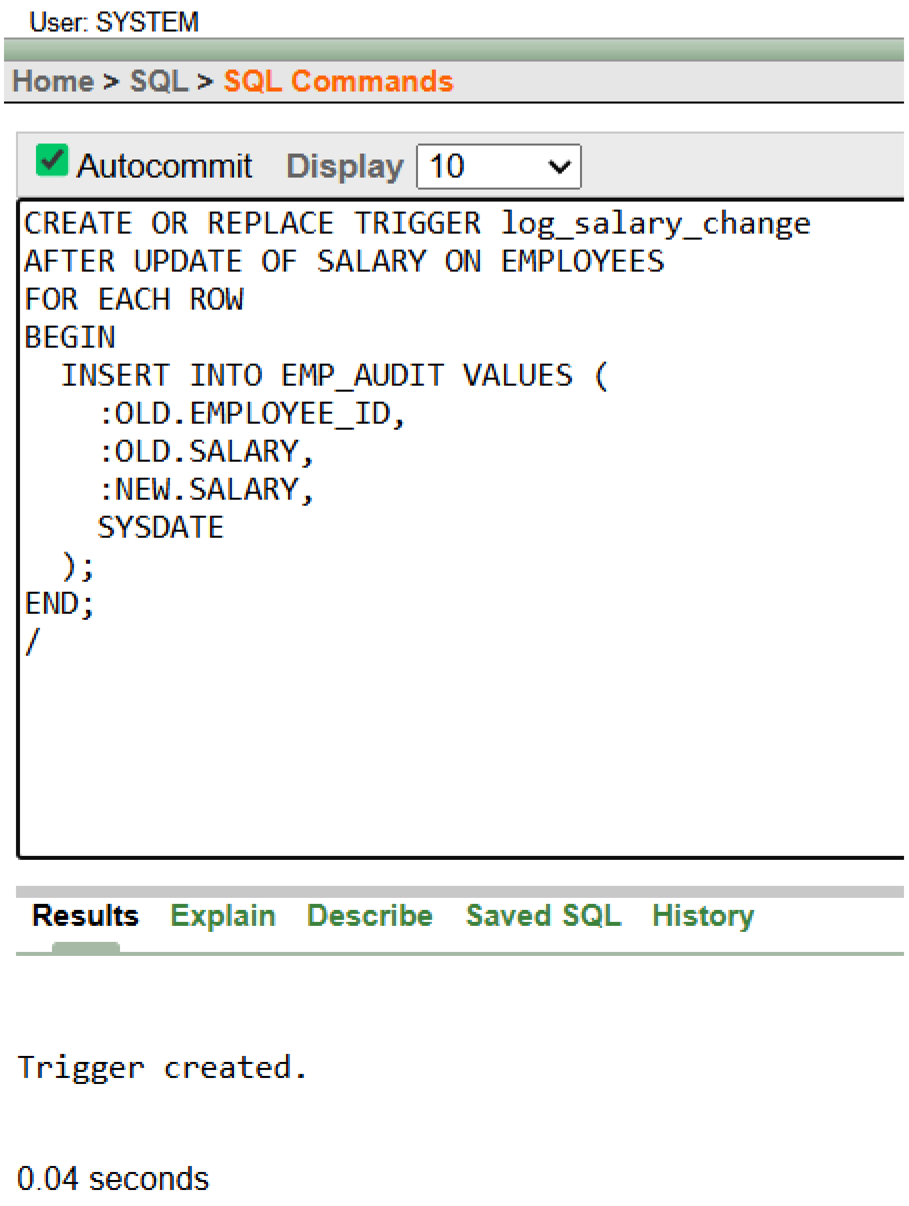
```
CREATE OR REPLACE TRIGGER restrict_total_sales
BEFORE INSERT ON SALES
FOR EACH ROW
DECLARE
  total NUMBER;
BEGIN
  SELECT NVL(SUM(AMOUNT), 0) INTO total FROM SALES;

  IF total + :NEW.AMOUNT > 100000 THEN
    RAISE_APPLICATION_ERROR(-20003, 'Sales limit exceeded.');
  END IF;
END;
/
```

```
Trigger created.
```

0.01 seconds

Tab 4

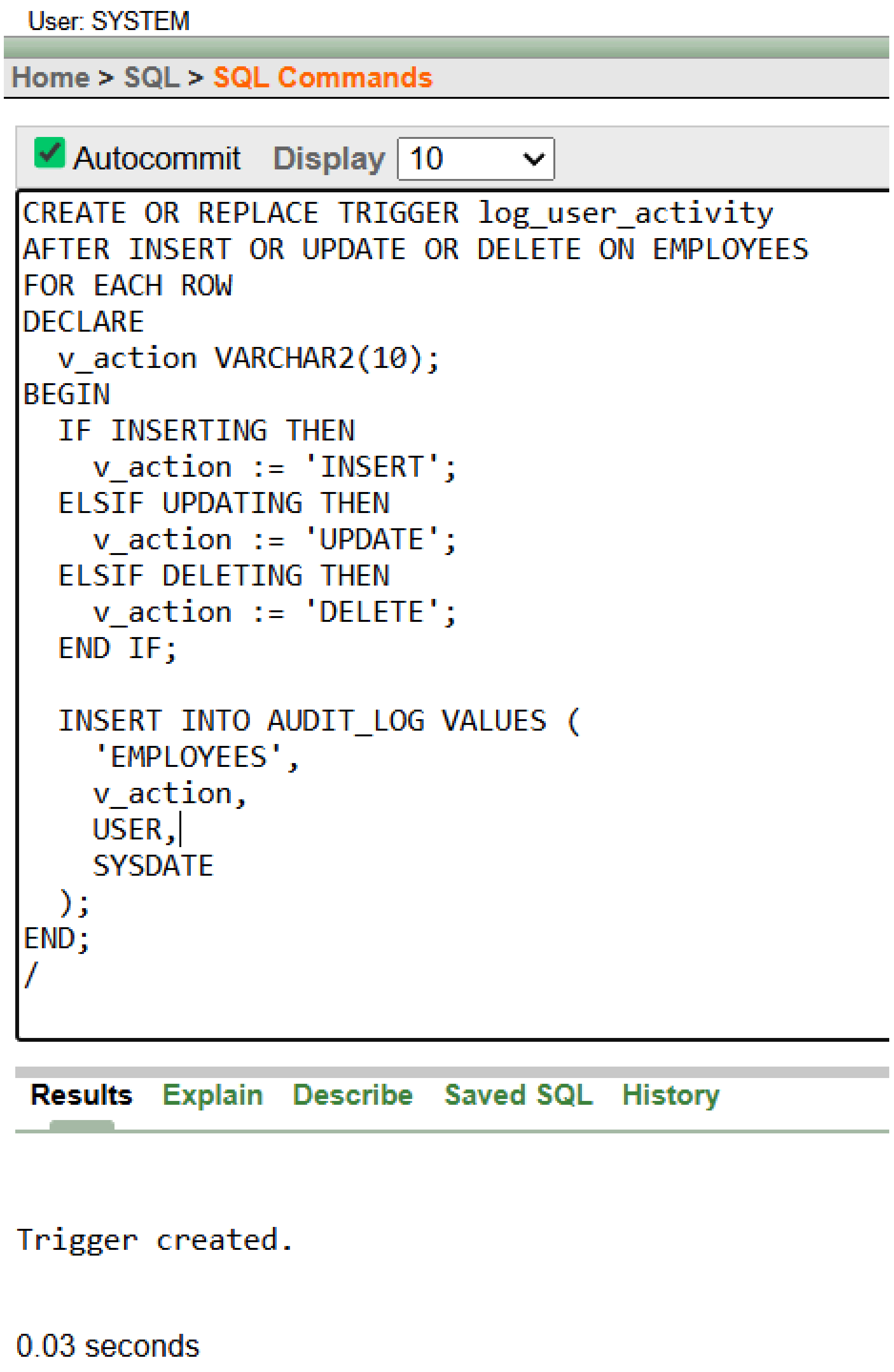


```
CREATE OR REPLACE TRIGGER log_salary_change
AFTER UPDATE OF SALARY ON EMPLOYEES
FOR EACH ROW
BEGIN
  INSERT INTO EMP_AUDIT VALUES (
    :OLD.EMPLOYEE_ID,
    :OLD.SALARY,
    :NEW.SALARY,
    SYSDATE
  );
END;
/
```

```
Trigger created.
```

0.04 seconds

Tab 5



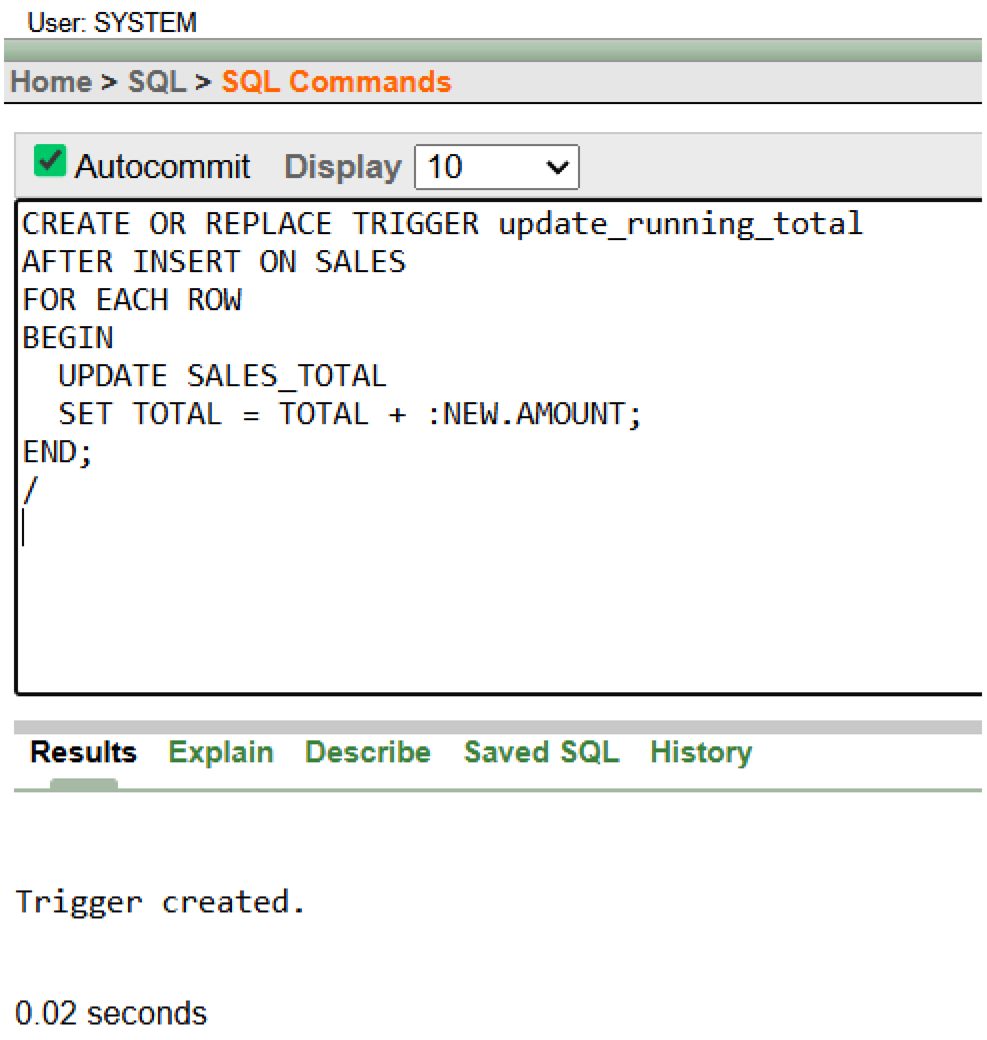
```
CREATE OR REPLACE TRIGGER log_user_activity
AFTER INSERT OR UPDATE OR DELETE ON EMPLOYEES
FOR EACH ROW
DECLARE
  v_action VARCHAR2(10);
BEGIN
  IF INSERTING THEN
    v_action := 'INSERT';
  ELSIF UPDATING THEN
    v_action := 'UPDATE';
  ELSIF DELETING THEN
    v_action := 'DELETE';
  END IF;

  INSERT INTO AUDIT_LOG VALUES (
    'EMPLOYEES',
    v_action,
    USER,
    SYSDATE
  );
END;
/
```

```
Trigger created.
```

0.03 seconds

Tab 6



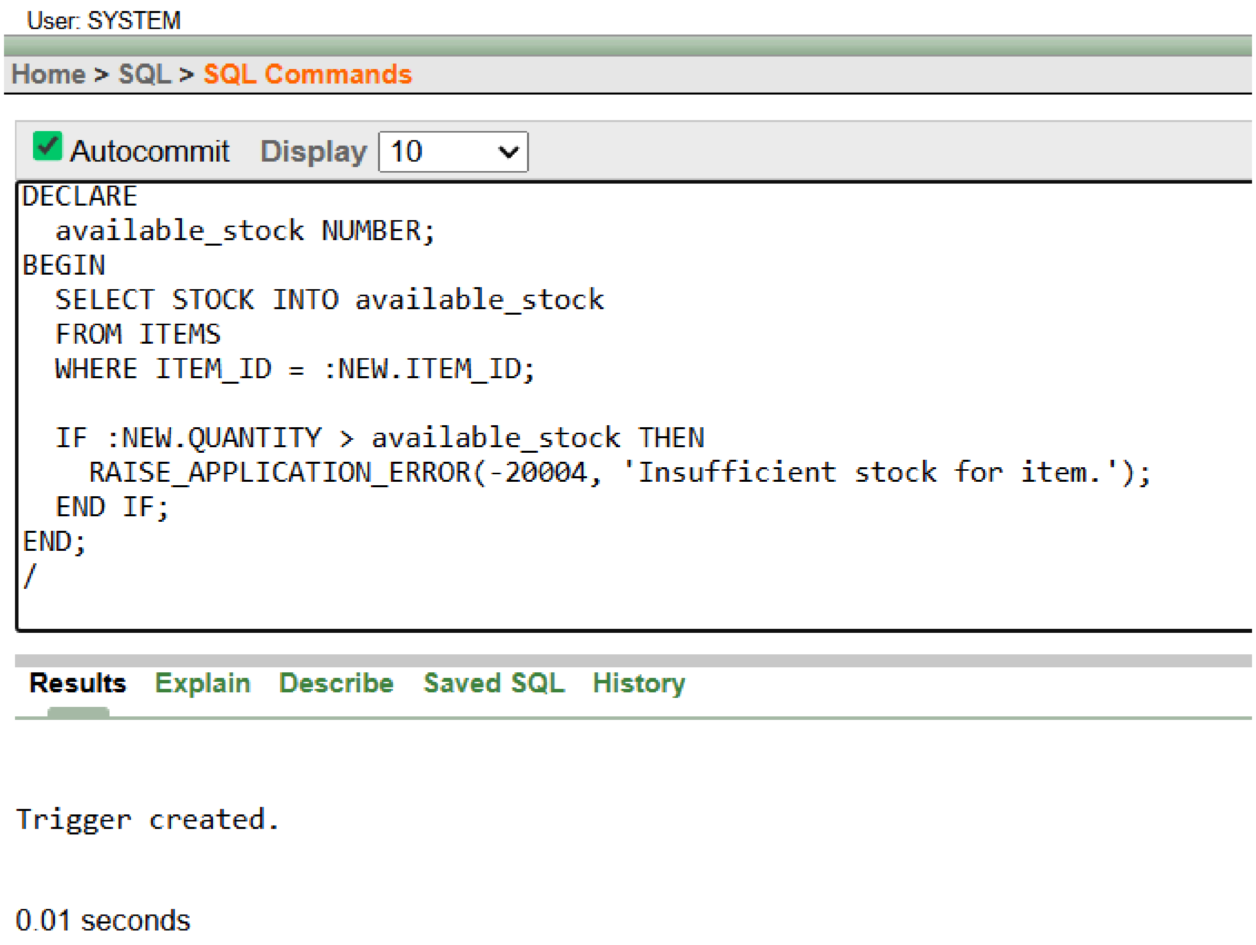
Autocommit ✓ Display 10

```
CREATE OR REPLACE TRIGGER update_running_total
AFTER INSERT ON SALES
FOR EACH ROW
BEGIN
  UPDATE SALES_TOTAL
  SET TOTAL = TOTAL + :NEW.AMOUNT;
END;
/
```

```
Trigger created.
```

0.02 seconds

Tab 7



User: SYSTEM

```
DECLARE
  available_stock NUMBER;
BEGIN
  SELECT STOCK INTO available_stock
  FROM ITEMS
  WHERE ITEM_ID = :NEW.ITEM_ID;

  IF :NEW.QUANTITY > available_stock THEN
    RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for item.');
  END IF;
END;
/
```

```
Trigger created.
```

0.01 seconds