Tabele:

```
CREATE TABLE firmy (
    firma_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    nazwa VARCHAR2(100) NOT NULL,
    adres VARCHAR2(200),
    telefon VARCHAR2(20),
    email VARCHAR2(100) UNIQUE NOT NULL
);

CREATE TABLE stanowiska (
    stanowisko_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    firma_id NUMBER NOT NULL,
    nazwa VARCHAR2(100) NOT NULL,
    opis VARCHAR2(500),
    wymagania VARCHAR2(500),
    wynagrodzenie NUMBER,
    data_dodania DATE DEFAULT SYSDATE,
    FOREIGN KEY (firma_id) REFERENCES firmy(firma_id) ON DELETE CASCADE
);

CREATE TABLE kandydaci (
    kandydat_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    imie VARCHAR2(50) NOT NULL,
    nazwisko VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    telefon VARCHAR2(20),
    pesel VARCHAR2(11) UNIQUE,
    data_aplikacji DATE DEFAULT SYSDATE
);

CREATE TABLE aplikacje (
    aplikacja_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    kandydat_id NUMBER NOT NULL,
    stanowisko_id NUMBER NOT NULL,
    data_aplikacji DATE DEFAULT SYSDATE,
    status VARCHAR2(20) DEFAULT 'W trakcie' CHECK (status IN ('W trakcie',
'Zaakceptowane', 'Odrzucone')),
    FOREIGN KEY (kandydat_id) REFERENCES kandydaci(kandydat_id) ON DELETE
CASCADE,
    FOREIGN KEY (stanowisko_id) REFERENCES stanowiska(stanowisko_id) ON DELETE
CASCADE
);
```

```
CREATE TABLE Logi (
    id_logu NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    akcja VARCHAR2(50),
    szczegoly VARCHAR2(2000),
    data_operacji DATE DEFAULT SYSDATE
);

CREATE TABLE archiwum_kandydaci AS SELECT * FROM kandydaci WHERE 1=0;

CREATE TABLE podsumowanie (
    id_podsumowania NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    typ_podsumowania VARCHAR2(20),
    okres VARCHAR2(10),
    liczba_aplikacji NUMBER,
    data_wygenerowania DATE DEFAULT SYSDATE
);
```

Funkcje:

```
create or replace FUNCTION liczba_aplikacji_kandydata (p_kandydat_id IN NUMBER)
RETURN NUMBER IS
   v_liczba_aplikacji NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_liczba_aplikacji
   FROM aplikacje
   WHERE kandydat_id = p_kandydat_id;

   RETURN v_liczba_aplikacji;
END;
/
CREATE OR REPLACE FUNCTION sprawdz_czy_aplikowal (p_kandydat_id IN NUMBER,
p_stanowisko_id IN NUMBER) RETURN NUMBER IS
   v_liczba_aplikacji NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_liczba_aplikacji
   FROM aplikacje
   WHERE kandydat_id = p_kandydat_id
     AND stanowisko_id = p_stanowisko_id;

   IF v_liczba_aplikacji > 0 THEN
      RETURN 1;
   ELSE
      RETURN 0;
   END IF;
END;
/
```

Procedury:

```
create or replace PROCEDURE aktualizuj_status_aplikacji (
    p_aplikacja_id IN NUMBER,
    p_status IN VARCHAR2
) IS
BEGIN
    UPDATE aplikacje
    SET status = p_status
    WHERE aplikacja_id = p_aplikacja_id;

    INSERT INTO Logi (akcja, szczegoly)
    VALUES ('Aktualizacja', 'Zmieniono status aplikacji ID: ' || p_aplikacja_id || ' na: ' ||
p_status);

    COMMIT;
END;
/

create or replace PROCEDURE dodaj_aplikacje (
    p_kandydat_id IN NUMBER,
    p_stanowisko_id IN NUMBER
) IS
BEGIN
    INSERT INTO aplikacje (kandydat_id, stanowisko_id, status)
    VALUES (p_kandydat_id, p_stanowisko_id, 'W trakcie');

    INSERT INTO Logi (akcja, szczegoly)
    VALUES ('Dodanie aplikacji', 'Kandydat ID: ' || p_kandydat_id || ' dodał aplikację na
stanowisko ID: ' || p_stanowisko_id);

    COMMIT;
END;
/
CREATE OR REPLACE PROCEDURE dodaj_kandydata (
    p_imie IN VARCHAR2,
    p_nazwisko IN VARCHAR2,
    p_email IN VARCHAR2,
    p_telefon IN VARCHAR2,
    p_pesel IN VARCHAR2
) IS
BEGIN
    IF NOT REGEXP_LIKE(p_pesel, '^\d{11}$') THEN
        RAISE_APPLICATION_ERROR(-20001, 'PESEL musi zawierać dokładnie 11 cyfr.');
    END IF;

    INSERT INTO kandydaci (imie, nazwisko, email, telefon, pesel)
```

```
    VALUES (p_imie, p_nazwisko, p_email, p_telefon, p_pesel);

    INSERT INTO Logi (akcja, szczegoly)
    VALUES ('Dodanie', 'Dodano kandydata: ' || p_imie || ' ' || p_nazwisko);

    COMMIT;
END;
/

create or replace PROCEDURE usun_kandydata (p_kandydat_id IN NUMBER) IS
BEGIN
    DELETE FROM kandydaci WHERE kandydat_id = p_kandydat_id;

    INSERT INTO Logi (akcja, szczegoly)
    VALUES ('Usunięcie', 'Usunięto kandydata ID: ' || p_kandydat_id);

    COMMIT;
END;
/

create or replace PROCEDURE podsumowanie_miesieczne (p_rok IN NUMBER, p_miesiac
IN NUMBER) IS
    v_liczba NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_liczba
    FROM aplikacje
    WHERE EXTRACT(YEAR FROM data_aplikacji) = p_rok
      AND EXTRACT(MONTH FROM data_aplikacji) = p_miesiac;

    INSERT INTO podsumowanie (typ_podsumowania, okres, liczba_aplikacji)
    VALUES ('Miesięczne', TO_CHAR(p_miesiac), v_liczba);

    COMMIT;
END;
/
create or replace PROCEDURE podsumowanie_kwartalne (p_rok IN NUMBER, p_kwartal
IN NUMBER) IS
    v_liczba NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_liczba
    FROM aplikacje
    WHERE EXTRACT(YEAR FROM data_aplikacji) = p_rok
      AND CEIL(EXTRACT(MONTH FROM data_aplikacji) / 3) = p_kwartal;

    INSERT INTO podsumowanie (typ_podsumowania, okres, liczba_aplikacji)
    VALUES ('Kwartalne', TO_CHAR(p_kwartal), v_liczba);

    COMMIT;
```

```
END;
/

create or replace PROCEDURE podsumowanie_roczne (p_rok IN NUMBER) IS
   v_liczba NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_liczba
   FROM aplikacje
   WHERE EXTRACT(YEAR FROM data_aplikacji) = p_rok;

   INSERT INTO podsumowanie (typ_podsumowania, okres, liczba_aplikacji)
   VALUES ('Roczne', TO_CHAR(p_rok), v_liczba);

   COMMIT;
END;
/
```

Triggery:

```
create or replace TRIGGER archiwizacja_kandydatow
BEFORE DELETE ON kandydaci
FOR EACH ROW
BEGIN
    INSERT INTO archiwum_kandydaci
    (kandydat_id, imie, nazwisko, email, telefon, pesel, data_aplikacji)
    VALUES (:OLD.kandydat_id, :OLD.imie, :OLD.nazwisko, :OLD.email, :OLD.telefon,
:OLD.pesel, :OLD.data_aplikacji);
END;
/

create or replace TRIGGER logowanie_kandydatow
AFTER INSERT OR UPDATE OR DELETE ON kandydaci
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO Logi (akcja, szczegoly) VALUES ('Dodanie', 'Dodano nowego kandydata:
' || :NEW.imie || ' ' || :NEW.nazwisko);
    ELSIF UPDATING THEN
        INSERT INTO Logi (akcja, szczegoly) VALUES ('Aktualizacja', 'Zaktualizowano
kandydata ID: ' || :OLD.kandydat_id);
    ELSIF DELETING THEN
        INSERT INTO Logi (akcja, szczegoly) VALUES ('Usuniecie', 'Usunieto kandydata ID: ' ||
:OLD.kandydat_id);
    END IF;
END;
/
```

Przykłady użycia:

Funkcje:
(id_kandydata, id aplikacji)
SELECT sprawdz_czy_aplikowal(32, 3) FROM dual;

(id kandydata)
SELECT liczba_aplikacji_kandydata(1) FROM dual;

Procedury:
(id_aplikacji, status)
EXEC aktualizuj_status_aplikacji(1, 'Zaakceptowane');

(id_kandydat, id_stanowisko)
EXEC dodaj_aplikacje(1, 2);

(imię, nazwisko, email, nr telefonu, pesel)
EXEC dodaj_kandydata('Jan', 'Kowalski', 'jan.kowalski@example.com', '123456789', '12345678901');

(id_kandydata)
EXEC usun_kandydata(1);

(rok, miesiąc)
EXEC podsumowanie_miesieczne(2025, 1);

(rok, kwartał)
EXEC podsumowanie_kwartalne(2025, 1);

(rok)
EXEC podsumowanie_roczne(2025);

Dodawanie ręczne wartości do tabel

firmy:

```
INSERT INTO firmy (nazwa, adres, telefon, email)
VALUES ('Firma A', 'ul. Przykładowa 35, Warszawa', '133-456-789', 'contactxd@firmaa.pl');

INSERT INTO firmy (nazwa, adres, telefon, email)
VALUES ('Firma B', 'ul. Przykładowa 2, Kraków', '987-654-321', 'contact@firmab.pl');

INSERT INTO firmy (nazwa, adres, telefon, email)
VALUES ('Firma C', 'ul. Przykładowa 3, Wrocław', '555-123-456', 'contact@firmac.pl');

INSERT INTO firmy (nazwa, adres, telefon, email)
VALUES ('Firma D', 'ul. Przykładowa 4, Poznań', '666-789-012', 'contact@firmad.pl');
```

stanowiska:

```
INSERT INTO stanowiska (firma_id, nazwa, opis, wymagania, wynagrodzenie)
VALUES (1, 'Programista', 'Tworzenie aplikacji webowych', 'Znajomość Pythona', 8000);

INSERT INTO stanowiska (firma_id, nazwa, opis, wymagania, wynagrodzenie)
VALUES (1, 'Analityk', 'Analiza danych', 'Znajomość SQL i Excel', 7000);


INSERT INTO stanowiska (firma_id, nazwa, opis, wymagania, wynagrodzenie)
VALUES (3, 'Projektant', 'Projektowanie UI/UX', 'Znajomość Figma', 7500);

INSERT INTO stanowiska (firma_id, nazwa, opis, wymagania, wynagrodzenie)
VALUES (4, 'Manager', 'Zarządzanie projektami', 'Doświadczenie w zarządzaniu', 9000);
```

kandydaci:

```
INSERT INTO kandydaci (imie, nazwisko, email, telefon, pesel)
VALUES ('Jan', 'Kowalski', 'jan.kowalski@example.com', '123-456-789', '12345678901');

INSERT INTO kandydaci (imie, nazwisko, email, telefon, pesel)
VALUES ('Anna', 'Nowak', 'anna.nowak@example.com', '987-654-321', '98765432101');

INSERT INTO kandydaci (imie, nazwisko, email, telefon, pesel)
VALUES ('Piotr', 'Zielinski', 'piotr.zielinski@example.com', '555-123-456', '55512345601');

INSERT INTO kandydaci (imie, nazwisko, email, telefon, pesel)
VALUES ('Katarzyna', 'Wójcik', 'katarzyna.wojcik@example.com', '666-789-012',
'66678901201');
```

Logi:

```
INSERT INTO Logi (akcja, szczegoly)
VALUES ('Dodanie', 'Dodano kandydata: Jan Kowalski');

INSERT INTO Logi (akcja, szczegoly)
VALUES ('Usuniecie', 'Usunięto kandydata ID: 1');

INSERT INTO Logi (akcja, szczegoly)
VALUES ('Aktualizacja', 'Zmieniono status aplikacji ID: 2 na Zaakceptowane');

INSERT INTO Logi (akcja, szczegoly)
VALUES ('Dodanie', 'Dodano stanowisko: Programista');
```

Archiwum kandydaci:

```
INSERT INTO archiwum_kandydaci
SELECT * FROM kandydaci WHERE kandydat_id IN (1, 2, 3, 4);
```

Podsumowanie:

```
INSERT INTO podsumowanie (typ_podsumowania, okres, liczba_aplikacji)
VALUES ('Miesieczne', '01', 10);

INSERT INTO podsumowanie (typ_podsumowania, okres, liczba_aplikacji)
VALUES ('Kwartalne', 'Q1', 25);

INSERT INTO podsumowanie (typ_podsumowania, okres, liczba_aplikacji)
VALUES ('Roczne', '2025', 100);

INSERT INTO podsumowanie (typ_podsumowania, okres, liczba_aplikacji)
VALUES ('Miesieczne', '02', 15);
```