3. TWORZENIE WIDOKÓW

3.1 wycieczki_osoby(kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

```
CREATE VIEW WYCIECZKI_OSOBY AS
  SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA,
    o.IMIE,
    o.NAZWISKO,
    r.STATUS
  FROM WYCIECZKI w
    JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
    JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
```

3.2 wycieczki_osoby_potwierdzone (kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

```
CREATE VIEW WYCIECZ_OSOBY_POTWIERDZONE_ORAZ_POTWIERDZONE_ZAPLACONE AS
 SELECT
  w.ID_WYCIECZKI,
  w.NAZWA,
  w.KRAJ,
  w.DATA,
  o.IMIE,
  o.NAZWISKO,
  r.STATUS
 FROM WYCIECZKI w
  JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
  JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
 WHERE r.STATUS = 'P' OR r.STATUS = 'Z'
```

3.3 wycieczki_przyszle (kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

```
CREATE VIEW WYCIECZKI_PRZYSZLE AS
  SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA,
    o.IMIE,
    o.NAZWISKO,
    r.STATUS
  FROM WYCIECZKI w
    JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
    JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
  WHERE w.DATA > CURRENT_DATE
```

3.4 wycieczki_miejsca(kraj,data, nazwa_wycieczki,liczba_miejsc, liczba_wolnych_miejsc)

```
CREATE VIEW WYCIECZKI_MIEJSCA_WOLNE_OK AS
  SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA,
```

```
    w.LICZBA_MIEJSC,
    w.LICZBA_MIEJSC - (SELECT count(ID_WYCIECZKI)
                       FROM REZERWACJE r
                       WHERE r.ID_WYCIECZKI = w.ID_WYCIECZKI AND r.STATUS
!= 'A') AS wolne
  FROM WYCIECZKI w
```

3.5 dostępne_wycieczki(kraj,data, nazwa_wycieczki,liczba_miejsc, liczba_wolnych_miejsc)

```
CREATE VIEW DOSTĘPNE_WYCIECZKI_OK AS
  SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA,
    w.LICZBA_MIEJSC - (SELECT count(ID_WYCIECZKI)
                       FROM REZERWACJE r
                       WHERE r.ID_WYCIECZKI = w.ID_WYCIECZKI AND r.STATUS
!= 'A') AS wolne
  FROM wycieczki_miejsca_wolne_ok w
  WHERE w.data > current_date AND w.wolne > 0
```

3.6 rezerwacje_do_ anulowania – lista niepotwierdzonych rezerwacji które powinne zostać anulowane, rezerwacje przygotowywane są do anulowania na tydzień przed wyjazdem)

```
CREATE VIEW REZERWACJE_DO_ANULOWANIA_OK AS
  SELECT
    r.NR_REZERWACJI,
    r.ID_WYCIECZKI,
    r.ID_OSOBY,
    r.STATUS
  FROM REZERWACJE r
    JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
  WHERE w.DATA - current_date <= 7 AND r.STATUS = 'N'
```

4. TWORZENIE PROCEDUR/FUNKCJI POBIERAJĄCYCH DANE.

4.1 uczestnicy_wycieczki (id_wycieczki), procedura ma zwracać podobny zestaw danych jak widok wycieczki_osoby

```
CREATE TYPE UCZESTNICY_WYCIECZKI_RECORD AS OBJECT (
  nazwa    VARCHAR2(100),
  kraj     VARCHAR2(50),
  data     DATE,
  imie     VARCHAR2(50),
  nazwisko VARCHAR2(50),
  status   CHAR(1)
)

CREATE TYPE UCZESTNICY_WYCIECZKI_TABLE AS TABLE OF
UCZESTNICY_WYCIECZKI_RECORD

CREATE FUNCTION uczestnicy_wycieczki(f_id_wycieczki IN NUMBER)
  RETURN UCZESTNICY_WYCIECZKI_TABLE
AS
  result   UCZESTNICY_WYCIECZKI_TABLE;
  id_exist NUMBER;
```

```
  BEGIN
    SELECT count(*)
    INTO id_exist
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = f_id_wycieczki;

    IF id_exist = 0
    THEN
      raise_application_error(-20001, 'Niestety wycieczka nie istnieje');
    END IF;


    SELECT uczestnicy_wycieczki_record(wo.NAZWA, wo.KRAJ, wo.DATA, wo.IMIE,
wo.NAZWISKO, wo.STATUS)
    BULK COLLECT INTO result
    FROM wycieczki_osoby wo
    WHERE wo.ID_WYCIECZKI = f_id_wycieczki;

    RETURN result;
  END;
```

4.2 rezerwacje_osoby(id_osoby), procedura ma zwracać podobny zestaw danych jak widok wycieczki_osoby

```
CREATE FUNCTION rezerwacje_osoby(f_id_osoby IN NUMBER)
  RETURN UCZESTNICY_WYCIECZKI_TABLE
AS
  result   UCZESTNICY_WYCIECZKI_TABLE;
  id_exist NUMBER;

  BEGIN
    SELECT count(*)
    INTO id_exist
    FROM OSOBY o
    WHERE o.ID_OSOBY = f_id_osoby;

    IF id_exist = 0
    THEN
      raise_application_error(-20001, 'Niestety osoba nie istnieje');
    END IF;


    SELECT uczestnicy_wycieczki_record(w.NAZWA, w.KRAJ, w.DATA, o.IMIE,
o.NAZWISKO, r.STATUS)
    BULK COLLECT INTO result
    FROM OSOBY o
      JOIN REZERWACJE r ON o.ID_OSOBY = r.ID_OSOBY
      JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
    WHERE o.ID_OSOBY = f_id_osoby;

    RETURN result;
  END;
```

4.3 przyszle_rezerwacje_osoby(id_osoby)

```
CREATE FUNCTION przyszle_rezerwacje_osoby(f_id_osoby IN NUMBER)
  RETURN UCZESTNICY_WYCIECZKI_TABLE
AS
  result   UCZESTNICY_WYCIECZKI_TABLE;
```

```
  id_exist NUMBER;

  BEGIN
    SELECT count(*)
    INTO id_exist
    FROM OSOBY o
    WHERE o.ID_OSOBY = f_id_osoby;

    IF id_exist = 0
    THEN
      raise_application_error(-20001, 'Niestety osoba nie istnieje');
    END IF;


    SELECT uczestnicy_wycieczki_record(w.NAZWA, w.KRAJ, w.DATA, o.IMIE,
o.NAZWISKO, r.STATUS)
    BULK COLLECT INTO result
    FROM OSOBY o
      JOIN REZERWACJE r ON o.ID_OSOBY = r.ID_OSOBY
      JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
    WHERE o.ID_OSOBY = f_id_osoby AND w.DATA > current_date;

    RETURN result;
  END;
```

4.4 dostepne_wycieczki(kraj, data_od, data_do)

```
CREATE TYPE WYCIECZKA_DOSTEPNA_RECORD AS OBJECT (
  id_wycieczki          INT,
  nazwa                 VARCHAR2(100),
  kraj                  VARCHAR2(50),
  data                  DATE,
  liczba_miejsc         INT,
  liczba_miejsc_wolnych INT
)

CREATE TYPE WYCIECZKA_DOSTEPNA_TABLE AS TABLE OF WYCIECZKA_DOSTEPNA_RECORD

CREATE FUNCTION dostepne_wycieczki_fun(f_kraj in VARCHAR2, data_od in DATE,
data_do in DATE)
  return wycieczka_dostepna_table
AS
  result wycieczka_dostepna_table;
  id_exist NUMBER;

  BEGIN
    SELECT count(*)
      INTO id_exist
      FROM WYCIECZKI w
      WHERE w.KRAJ = f_kraj;

    if id_exist = 0 THEN
      raise_application_error(-20001, 'Niestety wycieczka nie istnieje');
    END IF;


    SELECT wycieczka_dostepna_record(wmw.ID_WYCIECZKI, wmw.NAZWA, wmw.KRAJ,
wmw.DATA, wmw.LICZBA_MIEJSC, wmw.WOLNE)
    BULK COLLECT into result
```

```
    FROM WYCIECZKI_MIEJSCA_WOLNE_OK wmw
      where wmw.KRAJ = f_kraj AND  wmw.WOLNE > 0 and wmw.data BETWEEN
data_od and data_do;

    RETURN result;
  END;
```

5. TWORZENIE PROCEDUR MODYFIKUJĄCYCH DANE.

5.1 dodaj_rezerwacje(id_wycieczki, id_osoby)

```
CREATE PROCEDURE dodaj_rezerwacje(f_id_wycieczki IN NUMBER, f_id_osoby IN
NUMBER)
AS
  id_wycieczki_exist NUMBER;
  id_osoby_exist     NUMBER;


  BEGIN
    IF f_id_wycieczki IS NULL OR f_id_osoby IS NULL
    THEN
      raise_application_error(-20001, 'Podaj argumenty');
    END IF;

    SELECT count(*)
    INTO id_wycieczki_exist
    FROM DOSTĘPNE_WYCIECZKI_OK dw
    WHERE dw.ID_WYCIECZKI = f_id_wycieczki;

    SELECT count(*)
    INTO id_osoby_exist
    FROM OSOBY o
    WHERE o.ID_OSOBY = f_id_osoby;


    IF id_wycieczki_exist = 0
    THEN
      raise_application_error(-20000, 'wycieczka nie istnieje bądź już się
odbyła');
    END IF;
    IF id_osoby_exist = 0
    THEN
      raise_application_error(-20000, 'osoba nie istnieje');
    END IF;


    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (f_id_wycieczki, f_id_osoby, 'N');
    COMMIT;
  END;
```

5.2 zmien_status_rezerwacji(id_rezerwacji, status)

```
CREATE OR REPLACE PROCEDURE zmien_status_rezerwacji(f_id_rezerwacji IN
NUMBER, f_status IN CHAR)
AS
  id_rezerwacji_exist NUMBER;
  aktualny_status     CHAR(1);
  data_wycieczki      DATE;
```

```
  BEGIN
    SELECT count(*)
    INTO id_rezerwacji_exist
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF id_rezerwacji_exist = 0
    THEN
      raise_application_error(-20000, 'nie ma takiej rezerwacji');
    END IF;

    IF f_status NOT IN ('A', 'N', 'P', 'Z')
    THEN
      raise_application_error(-20000, 'podano zły status');
    END IF;

    SELECT w.data
    INTO data_wycieczki
    FROM REZERWACJE r
      JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF data_wycieczki < current_date
    THEN
      raise_application_error(-20000, 'wycieczka już się odbyła');
    END IF;


    SELECT r.STATUS
    INTO aktualny_status
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;


    IF (aktualny_status = 'A')
    THEN
      raise_application_error(-20000, 'nie można zmienić statusu anulowanej
rezerwacji');
    END IF;

    IF (aktualny_status = 'P' AND f_status = 'N')
    THEN
      raise_application_error(-20000, 'nie można z potwierdzonej na nową');
    END IF;

    IF (aktualny_status = 'Z' AND f_status IN ('N', 'P'))
    THEN
      raise_application_error(-20000, 'nie można z zapłaconej na
potwierdzoną albo nową');
    END IF;

    UPDATE REZERWACJE r
    SET r.STATUS = f_status
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;
  END;
```

5.3 zmien_liczbe_miejsc(id_wycieczki, liczba_miejsc)

```
CREATE PROCEDURE zmien_liczbe_miejsc(f_id_wycieczki IN NUMBER,
f_liczba_miejsc IN NUMBER)
AS
  id_wycieczki_exists NUMBER;
  liczba_zajetych     NUMBER;

  BEGIN

    SELECT count(*)
    INTO id_wycieczki_exists
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = f_id_wycieczki AND w.DATA > current_date;


    SELECT count(*)
    INTO liczba_zajetych
    FROM REZERWACJE r
    WHERE r.ID_WYCIECZKI = f_id_wycieczki AND r.STATUS != 'A';

    IF id_wycieczki_exists = 0
    THEN
      raise_application_error(-20000, 'wycieczka nie istnieje bądź się
odbyła');
    END IF;


    IF (f_liczba_miejsc - liczba_zajetych) < 0
    THEN
      raise_application_error(-20000, 'za mała wartość');
    END IF;

    UPDATE WYCIECZKI w
    SET w.LICZBA_MIEJSC = f_liczba_miejsc
    WHERE w.ID_WYCIECZKI = f_id_wycieczki;
  END;
```

6. DODAJEMY TABLICĘ DZIENNIKUJĄCĄ ZMIANY STATUSU REZERWACJI

```
CREATE TABLE REZERWACJE_LOG
(
  ID            NUMBER GENERATED AS IDENTITY,
  ID_REZERWACJI NUMBER,
  DATA          DATE,
  STATUS        CHAR
)

CREATE PROCEDURE dodaj_rezerwacje_log(f_id_wycieczki IN NUMBER, f_id_osoby
IN NUMBER)
AS
  id_wycieczki_exist NUMBER;
  id_osoby_exist     NUMBER;
  id_rezerwacji      NUMBER;

  BEGIN
    IF f_id_wycieczki IS NULL OR f_id_osoby IS NULL
    THEN
      raise_application_error(-20001, 'Podaj argumenty');
    END IF;
```

```
    SELECT count(*)
    INTO id_wycieczki_exist
    FROM DOSTĘPNE_WYCIECZKI_OK dw
    WHERE dw.ID_WYCIECZKI = f_id_wycieczki;

    SELECT count(*)
    INTO id_osoby_exist
    FROM OSOBY o
    WHERE o.ID_OSOBY = f_id_osoby;


    IF id_wycieczki_exist = 0
    THEN
      raise_application_error(-20000, 'wycieczka nie istnieje bądź już się
odbyła');
    END IF;
    IF id_osoby_exist = 0
    THEN
      raise_application_error(-20000, 'osoba nie istnieje');
    END IF;


    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (f_id_wycieczki, f_id_osoby, 'N')
    RETURNING NR_REZERWACJI INTO id_rezerwacji;
    COMMIT;

    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (id_rezerwacji, current_date, 'N');
    COMMIT;

  END;


CREATE OR REPLACE PROCEDURE zmien_status_rezerwacji_2(f_id_rezerwacji IN
NUMBER, f_status IN CHAR)
AS
  id_rezerwacji_exist NUMBER;
  aktualny_status     CHAR(1);
  data_wycieczki      DATE;


  BEGIN
    SELECT count(*)
    INTO id_rezerwacji_exist
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF id_rezerwacji_exist = 0
    THEN
      raise_application_error(-20000, 'nie ma takiej rezerwacji');
    END IF;

    IF f_status NOT IN ('A', 'N', 'P', 'Z')
    THEN
      raise_application_error(-20000, 'podano zły status');
    END IF;

    SELECT w.data
```

```
    INTO data_wycieczki
    FROM REZERWACJE r
      JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF data_wycieczki < current_date
    THEN
      raise_application_error(-20000, 'wycieczka już się odbyła');
    END IF;


    SELECT r.STATUS
    INTO aktualny_status
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;


    IF (aktualny_status = 'A')
    THEN
      raise_application_error(-20000, 'nie można zmienić statusu anulowanej
rezerwacji');
    END IF;

    IF (aktualny_status = 'P' AND f_status = 'N')
    THEN
      raise_application_error(-20000, 'nie można z potwierdzonej na nową');
    END IF;

    IF (aktualny_status = 'Z' AND f_status IN ('N', 'P'))
    THEN
      raise_application_error(-20000, 'nie można z zapłaconej na
potwierdzoną albo nową');
    END IF;

    UPDATE REZERWACJE r
    SET r.STATUS = f_status
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (f_id_rezerwacji, current_date, f_status);
  END;
```

7. ZMIANA STRUKTURY BAZY DANYCH, W TABELI WYCIECZKI DODAJEMY REDUNDANTNE POLE LICZBA_WOLNYCH_MIEJSC

```
ALTER TABLE WYCIECZKI
  ADD liczba_wolnych_miejsc INT;

CREATE PROCEDURE oblicz_wolne_miejsca
AS
  BEGIN
    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = (w.LICZBA_MIEJSC - (SELECT count(*)
                                                       FROM REZERWACJE r
                                                       WHERE r.ID_WYCIECZKI
= w.ID_WYCIECZKI AND r.STATUS != 'A'));
  END;
```

```
BEGIN
  oblicz_wolne_miejsca();
END;


CREATE VIEW WYCIECZKI_MIEJSCA_WOLNE_2 AS
  SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA,
    w.LICZBA_MIEJSC,
    w.liczba_wolnych_miejsc
  FROM WYCIECZKI w


CREATE VIEW DOSTĘPNE_WYCIECZKI__2 AS
  SELECT
    w.ID_WYCIECZKI,
    w.NAZWA,
    w.KRAJ,
    w.DATA,
    w.liczba_wolnych_miejsc
  FROM wycieczki w
  WHERE w.data > current_date AND w.LICZBA_WOLNYCH_MIEJSC > 0


CREATE FUNCTION dostepne_wycieczki_fun_2(f_kraj IN VARCHAR2, data_od IN
DATE, data_do IN DATE)
  RETURN WYCIECZKA_DOSTEPNA_TABLE
AS
  result   WYCIECZKA_DOSTEPNA_TABLE;
  id_exist NUMBER;

  BEGIN
    SELECT count(*)
    INTO id_exist
    FROM WYCIECZKI w
    WHERE w.KRAJ = f_kraj;

    IF id_exist = 0
    THEN
      raise_application_error(-20001, 'Niestety wycieczka nie istnieje');
    END IF;


    SELECT wycieczka_dostepna_record(wmw.ID_WYCIECZKI, wmw.NAZWA, wmw.KRAJ,
wmw.DATA, wmw.LICZBA_MIEJSC,
                                     wmw.LICZBA_WOLNYCH_MIEJSC)
    BULK COLLECT INTO result
    FROM WYCIECZKI w
    WHERE w.KRAJ = f_kraj AND wmw.WOLNE > 0 AND wmw.data BETWEEN data_od
AND data_do;

    RETURN result;
  END;
```

```
CREATE PROCEDURE dodaj_rezerwacje_2(f_id_wycieczki IN NUMBER, f_id_osoby IN
NUMBER)
AS
  id_wycieczki_exist NUMBER;
  id_osoby_exist     NUMBER;


  BEGIN
    IF f_id_wycieczki IS NULL OR f_id_osoby IS NULL
    THEN
      raise_application_error(-20001, 'Podaj argumenty');
    END IF;

    SELECT count(*)
    INTO id_wycieczki_exist
    FROM DOSTĘPNE_WYCIECZKI_OK dw
    WHERE dw.ID_WYCIECZKI = f_id_wycieczki;

    SELECT count(*)
    INTO id_osoby_exist
    FROM OSOBY o
    WHERE o.ID_OSOBY = f_id_osoby;


    IF id_wycieczki_exist = 0
    THEN
      raise_application_error(-20000, 'wycieczka nie istnieje bądź już się
odbyła');
    END IF;
    IF id_osoby_exist = 0
    THEN
      raise_application_error(-20000, 'osoba nie istnieje');
    END IF;

    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (f_id_wycieczki, f_id_osoby, 'N');
    COMMIT;

    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC - 1
    WHERE f_id_wycieczki = w.ID_WYCIECZKI;
    COMMIT;
  END;


CREATE OR REPLACE PROCEDURE zmien_status_rezerwacji_3(f_id_rezerwacji IN
NUMBER, f_status IN CHAR)
AS
  id_rezerwacji_exist NUMBER;
  aktualny_status     CHAR(1);
  data_wycieczki      DATE;


  BEGIN
    SELECT count(*)
    INTO id_rezerwacji_exist
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF id_rezerwacji_exist = 0
```

```
    THEN
      raise_application_error(-20000, 'nie ma takiej rezerwacji');
    END IF;

    IF f_status NOT IN ('A', 'N', 'P', 'Z')
    THEN
      raise_application_error(-20000, 'podano zły status');
    END IF;

    SELECT w.data
    INTO data_wycieczki
    FROM REZERWACJE r
      JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF data_wycieczki < current_date
    THEN
      raise_application_error(-20000, 'wycieczka już się odbyła');
    END IF;


    SELECT r.STATUS
    INTO aktualny_status
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;


    IF (aktualny_status = 'A')
    THEN
      raise_application_error(-20000, 'nie można zmienić statusu anulowanej
rezerwacji');
    END IF;

    IF (aktualny_status = 'P' AND f_status = 'N')
    THEN
      raise_application_error(-20000, 'nie można z potwierdzonej na nową');
    END IF;

    IF (aktualny_status = 'Z' AND f_status IN ('N', 'P'))
    THEN
      raise_application_error(-20000, 'nie można z zapłaconej na
potwierdzoną albo nową');
    END IF;

    UPDATE REZERWACJE r
    SET r.STATUS = f_status
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (f_id_rezerwacji, current_date, f_status);

    IF f_status = 'A'
    THEN
      UPDATE WYCIECZKI w
      SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC + 1
      WHERE w.ID_WYCIECZKI = (SELECT r.ID_WYCIECZKI
                              FROM REZERWACJE r
                              WHERE r.NR_REZERWACJI = f_id_rezerwacji);
    END IF;
  END;
```

```
CREATE PROCEDURE zmien_liczbe_miejsc_2(f_id_wycieczki IN NUMBER,
f_liczba_miejsc IN NUMBER)
AS
  id_wycieczki_exists NUMBER;
  liczba_zajetych     NUMBER;

  BEGIN

    SELECT count(*)
    INTO id_wycieczki_exists
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = f_id_wycieczki AND w.DATA > current_date;


    SELECT count(*)
    INTO liczba_zajetych
    FROM REZERWACJE r
    WHERE r.ID_WYCIECZKI = f_id_wycieczki AND r.STATUS != 'A';

    IF id_wycieczki_exists = 0
    THEN
      raise_application_error(-20000, 'wycieczka nie istnieje bądź się
odbyła');
    END IF;


    IF (f_liczba_miejsc - liczba_zajetych) < 0
    THEN
      raise_application_error(-20000, 'za mała wartość');
    END IF;

    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC +
(f_liczba_miejsc - w.LICZBA_MIEJSC)
    WHERE w.ID_WYCIECZKI = f_id_wycieczki;

    UPDATE WYCIECZKI w
    SET w.LICZBA_MIEJSC = f_liczba_miejsc
    WHERE w.ID_WYCIECZKI = f_id_wycieczki;

  END;
```

8. ZMIANA STRATEGII ZAPISYWANIA DO DZIENNIKA REZERWACJI. REALIZACJA PRZY POMOCY TRIGGERÓW.

```
CREATE OR REPLACE TRIGGER dodanie_zmiana_rezerwacje_log_trigger
AFTER INSERT OR UPDATE ON REZERWACJE
FOR EACH ROW
WHEN (NEW.NR_REZERWACJI > 0)
  BEGIN
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, rezerwacje_log.data,
rezerwacje_log.STATUS)
    VALUES (:NEW.NR_REZERWACJI, current_date, :new.STATUS);
  END;
```

```sql
CREATE OR REPLACE TRIGGER usuniecie_rezerwacje_log_trigger
BEFORE DELETE ON REZERWACJE
  BEGIN
    RAISE_APPLICATION_ERROR(-20001, 'nie wolno usuwać rezerwacji');
  END;


CREATE PROCEDURE dodaj_rezerwacje_3(f_id_wycieczki IN NUMBER, f_id_osoby IN
NUMBER)
AS
  id_wycieczki_exist NUMBER;
  id_osoby_exist     NUMBER;


  BEGIN
    IF f_id_wycieczki IS NULL OR f_id_osoby IS NULL
    THEN
      raise_application_error(-20001, 'Podaj argumenty');
    END IF;

    SELECT count(*)
    INTO id_wycieczki_exist
    FROM DOSTĘPNE_WYCIECZKI_OK dw
    WHERE dw.ID_WYCIECZKI = f_id_wycieczki;

    SELECT count(*)
    INTO id_osoby_exist
    FROM OSOBY o
    WHERE o.ID_OSOBY = f_id_osoby;


    IF id_wycieczki_exist = 0
    THEN
      raise_application_error(-20000, 'wycieczka nie istnieje bądź już się
odbyła');
    END IF;
    IF id_osoby_exist = 0
    THEN
      raise_application_error(-20000, 'osoba nie istnieje');
    END IF;


    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (f_id_wycieczki, f_id_osoby, 'N');
    COMMIT;

    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC - 1
    WHERE f_id_wycieczki = w.ID_WYCIECZKI;
    COMMIT;
  END;


CREATE OR REPLACE PROCEDURE zmien_status_rezerwacji_4(f_id_rezerwacji IN
NUMBER, f_status IN CHAR)
AS
  id_rezerwacji_exist NUMBER;
  aktualny_status     CHAR(1);
  data_wycieczki      DATE;
```

```
  BEGIN
    SELECT count(*)
    INTO id_rezerwacji_exist
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF id_rezerwacji_exist = 0
    THEN
      raise_application_error(-20000, 'nie ma takiej rezerwacji');
    END IF;

    IF f_status NOT IN ('A', 'N', 'P', 'Z')
    THEN
      raise_application_error(-20000, 'podano zły status');
    END IF;

    SELECT w.data
    INTO data_wycieczki
    FROM REZERWACJE r
      JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF data_wycieczki < current_date
    THEN
      raise_application_error(-20000, 'wycieczka już się odbyła');
    END IF;


    SELECT r.STATUS
    INTO aktualny_status
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;


    IF (aktualny_status = 'A')
    THEN
      raise_application_error(-20000, 'nie można zmienić statusu anulowanej
rezerwacji');
    END IF;

    IF (aktualny_status = 'P' AND f_status = 'N')
    THEN
      raise_application_error(-20000, 'nie można z potwierdzonej na nową');
    END IF;

    IF (aktualny_status = 'Z' AND f_status IN ('N', 'P'))
    THEN
      raise_application_error(-20000, 'nie można z zapłaconej na
potwierdzoną albo nową');
    END IF;

    UPDATE REZERWACJE r
    SET r.STATUS = f_status
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF f_status = 'A'
    THEN
      UPDATE WYCIECZKI w
```

```
      SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC + 1
      WHERE w.ID_WYCIECZKI = (SELECT r.ID_WYCIECZKI
                              FROM REZERWACJE r
                              WHERE r.NR_REZERWACJI = f_id_rezerwacji);
    END IF;
  END;
```

9. ZMIANA STRATEGII OBSŁUGI REDUNDANTNEGO POLA LICZBA_WOLNYCH_MIEJSC. REALIZACJA PRZY POMOCY TRIGGERÓW.

```
CREATE OR REPLACE TRIGGER dodanie_liczba_wolnych_miejsc_trigger
AFTER INSERT ON REZERWACJE
FOR EACH ROW
WHEN (NEW.STATUS IN ('N', 'P', 'Z'))
  BEGIN
    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC - 1
    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
  END;


CREATE OR REPLACE TRIGGER zmiana_liczba_wolnych_miejsc_trigger
AFTER UPDATE OF STATUS
  ON REZERWACJE
FOR EACH ROW
WHEN (NEW.STATUS = 'A' AND OLD.STATUS <> 'A')
  BEGIN
    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC + 1
    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
  END;


CREATE OR REPLACE TRIGGER zmiana_liczba_wolnych_miejsc_trigger_2
BEFORE UPDATE OF STATUS
  ON REZERWACJE
FOR EACH ROW
WHEN (NEW.STATUS <> 'A' AND OLD.STATUS = 'A')
  BEGIN
    RAISE_APPLICATION_ERROR(-20001, 'nie wolno zmieniać statusu anulowanej
rezerwacji');
  END;


CREATE OR REPLACE TRIGGER wycieczka_liczba_wolnych_miejsc_trigger
AFTER UPDATE OF LICZBA_MIEJSC
  ON WYCIECZKI
FOR EACH ROW
WHEN (NEW.LICZBA_MIEJSC <> OLD.LICZBA_MIEJSC)
  BEGIN
    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC +
(:NEW.LICZBA_MIEJSC - :OLD.LICZBA_MIEJSC)
    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
  END;
```

```
CREATE PROCEDURE dodaj_rezerwacje_4(f_id_wycieczki IN NUMBER, f_id_osoby IN
NUMBER)
AS
  id_wycieczki_exist NUMBER;
  id_osoby_exist     NUMBER;


  BEGIN
    IF f_id_wycieczki IS NULL OR f_id_osoby IS NULL
    THEN
      raise_application_error(-20001, 'Podaj argumenty');
    END IF;

    SELECT count(*)
    INTO id_wycieczki_exist
    FROM DOSTĘPNE_WYCIECZKI_OK dw
    WHERE dw.ID_WYCIECZKI = f_id_wycieczki;

    SELECT count(*)
    INTO id_osoby_exist
    FROM OSOBY o
    WHERE o.ID_OSOBY = f_id_osoby;


    IF id_wycieczki_exist = 0
    THEN
      raise_application_error(-20000, 'wycieczka nie istnieje bądź już się
odbyła');
    END IF;
    IF id_osoby_exist = 0
    THEN
      raise_application_error(-20000, 'osoba nie istnieje');
    END IF;


    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (f_id_wycieczki, f_id_osoby, 'N');
    COMMIT;

  END;


CREATE PROCEDURE zmien_liczbe_miejsc_4(f_id_wycieczki IN NUMBER,
f_liczba_miejsc IN NUMBER)
AS
  id_wycieczki_exists NUMBER;
  liczba_zajetych     NUMBER;

  BEGIN

    SELECT count(*)
    INTO id_wycieczki_exists
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = f_id_wycieczki AND w.DATA > current_date;


    SELECT count(*)
    INTO liczba_zajetych
    FROM REZERWACJE r
    WHERE r.ID_WYCIECZKI = f_id_wycieczki AND r.STATUS != 'A';
```

```
    IF id_wycieczki_exists = 0
    THEN
      raise_application_error(-20000, 'wycieczka nie istnieje bądź się
odbyła');
    END IF;


    IF (f_liczba_miejsc - liczba_zajetych) < 0
    THEN
      raise_application_error(-20000, 'za mała wartość');
    END IF;

    UPDATE WYCIECZKI w
    SET w.LICZBA_MIEJSC = f_liczba_miejsc
    WHERE w.ID_WYCIECZKI = f_id_wycieczki;

  END;


  CREATE OR REPLACE PROCEDURE zmien_status_rezerwacji_5(f_id_rezerwacji IN
NUMBER, f_status IN CHAR)
AS
  id_rezerwacji_exist NUMBER;
  aktualny_status     CHAR(1);
  data_wycieczki      DATE;


  BEGIN
    SELECT count(*)
    INTO id_rezerwacji_exist
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF id_rezerwacji_exist = 0
    THEN
      raise_application_error(-20000, 'nie ma takiej rezerwacji');
    END IF;

    IF f_status NOT IN ('A', 'N', 'P', 'Z')
    THEN
      raise_application_error(-20000, 'podano zły status');
    END IF;

    SELECT w.data
    INTO data_wycieczki
    FROM REZERWACJE r
      JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

    IF data_wycieczki < current_date
    THEN
      raise_application_error(-20000, 'wycieczka już się odbyła');
    END IF;


    SELECT r.STATUS
    INTO aktualny_status
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;
```

```
    IF (aktualny_status = 'A')
    THEN
      raise_application_error(-20000, 'nie można zmienić statusu anulowanej
rezerwacji');
    END IF;

    IF (aktualny_status = 'P' AND f_status = 'N')
    THEN
      raise_application_error(-20000, 'nie można z potwierdzonej na nową');
    END IF;

    IF (aktualny_status = 'Z' AND f_status IN ('N', 'P'))
    THEN
      raise_application_error(-20000, 'nie można z zapłaconej na
potwierdzoną albo nową');
    END IF;

    UPDATE REZERWACJE r
    SET r.STATUS = f_status
    WHERE r.NR_REZERWACJI = f_id_rezerwacji;

  END;
```