

WOJSKOWA AKADEMIA TECHNICZNA

# Bazy Danych

Dokumentacja projektu

Prowadzący: Mgr inż. Józef Woźniak

Wykonał: Sebastian Buczek

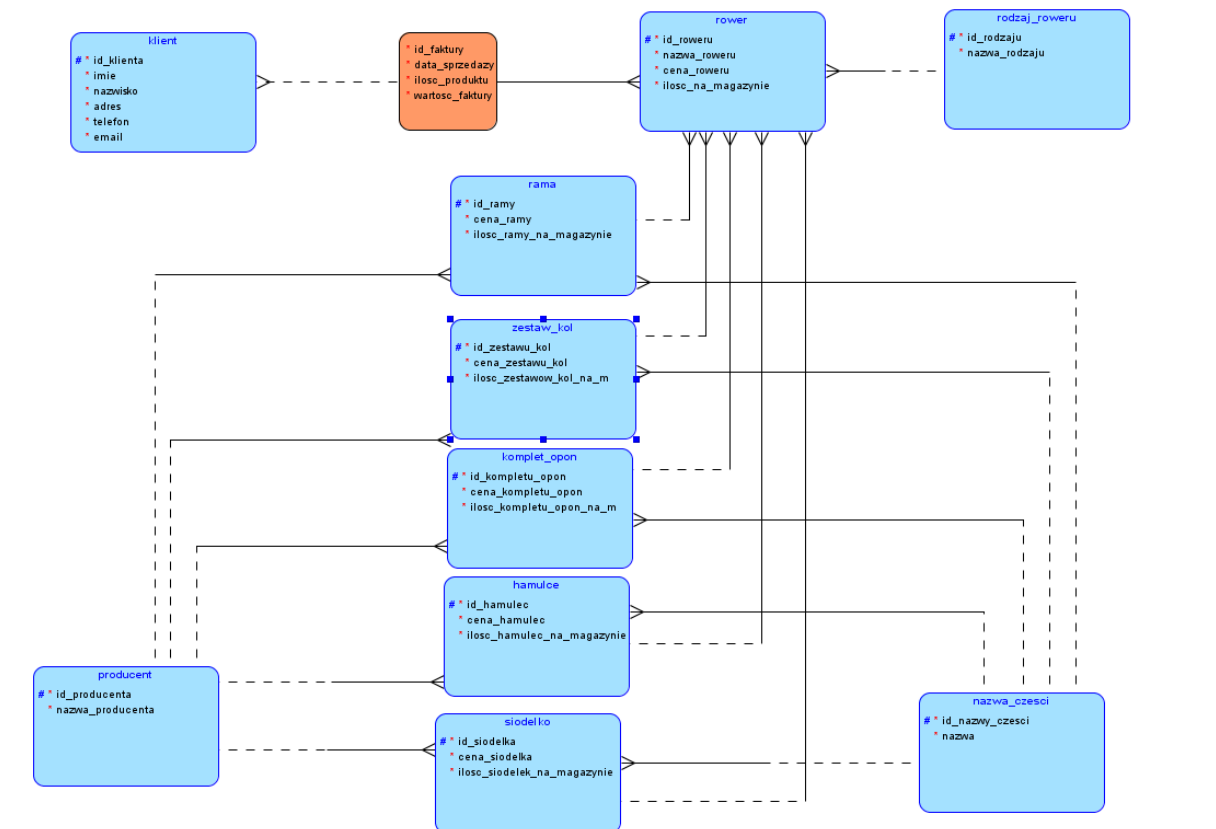
Grupa: I7Y1S1

Nr.indeksu: 69235

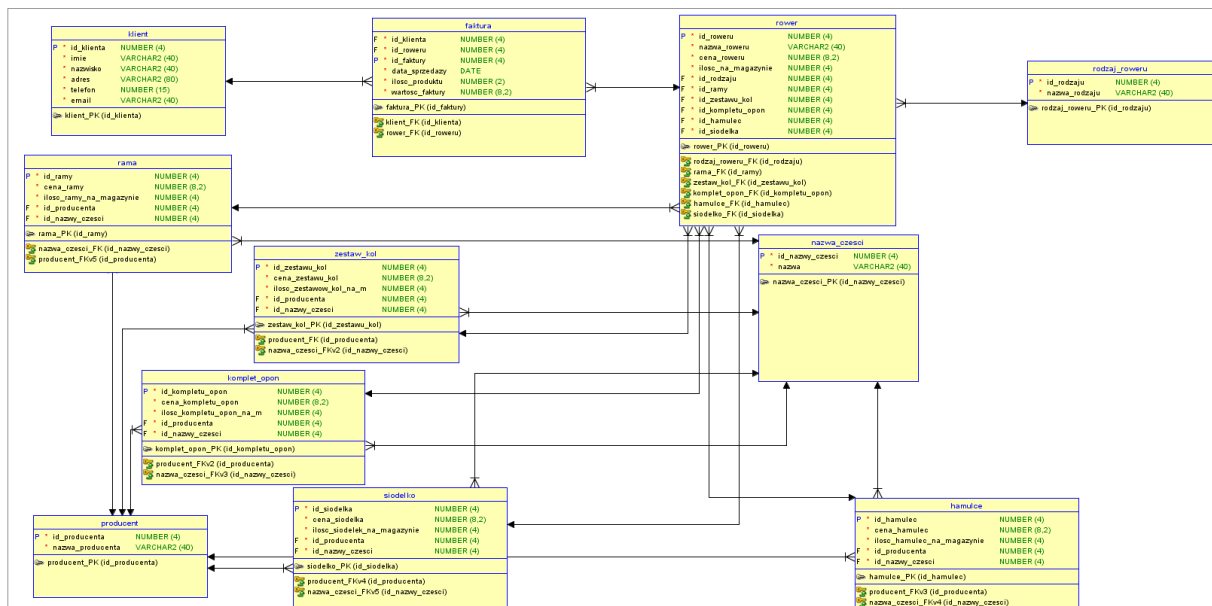
Nr. Konta na serwerze: IBD\_01

Tematyka mojego projektu zaliczeniowego to sklep rowerowy w którym klient może dowolnie skonfigurować swój rower lub kupić jeden z gotowych egzemplarzy. Cały system podlega fakturowaniu. Rower w moim założeniu składa się z 5 elementów: ramy, zestawu kół, kompletu opon, hamulców oraz z siodełka. Wszystkie te elementy mają swoją oddzielną tablicę w mojej bazie. Rower ma swój rodzaj co pozwala skategoryzować magazyn. Wszystkie części mogą mieć wspólnego lub różnych producentów. Jest to odwzorowaniem rzeczywistości. Każda część ma też swój ogólny model pomagający określić prestiż części lub też zwykłą solidność (tabela nazwa\_części(lekko nietrafiona nazwa)). Wykonany projekt obsługuje wiele zadań związanych z prowadzeniem podobnej działalności w rzeczywistości. Zapraszam do zapoznania się z dalszą częścią dokumentacji.

Model logiczny:



## Model relacyjny:



## Wyzwalacze odpowiedzialne za spójność bazy danych:

- Wyzwalacz zmieniający ilość rowerów w magazynie po ich sprzedaży
 

```

      create or replace trigger tr_update_sold_rower
      before insert on faktura
      for each row
      begin
      update rower
      set ilosc_na_magazynie = ilosc_na_magazynie - :NEW.ilosc_produkta
      where rower.id_roweru = :NEW.id_roweru;
      end tr_update_sold_rower;
      
```
- Wyzwalacz aktualizujący liczbę ram na magazynie po zbudowaniu roweru z ich użyciem
 

```

      create or replace trigger tr_update_sold_rama
      before insert on rower
      for each row
      begin
      update rama
      
```

```

set ilosc_razy_na_magazynie = ilosc_razy_na_magazynie -:NEW.ilosc_na_magazynie

where rama.id_razy = :NEW.id_razy;

end tr_update_sold_razy;

```

- Wyzwalacz aktualizujący liczbę hamulców na magazynie po zbudowaniu roweru z ich użyciem

```

create or replace trigger tr_update_sold_hamulec
before insert on rower
for each row
begin
update hamulce
set ilosc_hamulec_na_magazynie = ilosc_hamulec_na_magazynie -:NEW.ilosc_na_magazynie
where hamulce.id_hamulec = :NEW.id_hamulec;
end tr_update_sold_hamulec;

```

- Wyzwalacz aktualizujący liczbę siodełek na magazynie po zbudowaniu roweru z ich użyciem

```

create or replace trigger tr_update_sold_siodelko
before insert on rower
for each row
begin
update siodelko
set ilosc_siodelek_na_magazynie = ilosc_siodelek_na_magazynie -:NEW.ilosc_na_magazynie
where siodelko.id_siodelka = :NEW.id_siodelka;
end tr_update_sold_siodelko;

```

- Wyzwalacz aktualizujący liczbę zestawów kół na magazynie po zbudowaniu roweru z ich użyciem

```

create or replace trigger tr_update_sold_zestaw_kol
before insert on rower
for each row
begin
update zestaw_kol
set ilosc_zestawow_kol_na_m = ilosc_zestawow_kol_na_m -:NEW.ilosc_na_magazynie
where zestaw_kol.id_zestawu_kol = :NEW.id_zestawu_kol;
end tr_update_sold_zestaw_kol;

```

- Wyzwalacz aktualizujący liczbę kompletów opon na magazynie po zbudowaniu roweru z ich użyciem

```
create or replace trigger tr_update_sold_komplet_opon
before insert on rower
for each row
begin
update komplet_opon
set ilosc_kompletu_opon_na_m = ilosc_kompletu_opon_na_m - :NEW.ilosc_na_magazynie
where komplet_opon.id_kompletu_opon = :NEW.id_kompletu_opon;
end tr_update_sold_komplet_opon;
```

#### FUNKCJE:

- Funkcja sprawdzająca czy w bazie znajduje się już dany rodzaj roweru

```
create or replace function fn_rodzaj_check
(
v_nazwa_rodzaju in varchar2
)return boolean as v_bool number;
begin
select count (nazwa_rodzaju) into v_bool from rodzaj_roweru
where nazwa_rodzaju=v_nazwa_rodzaju;
if(v_bool=1) then return true;
else return false;
end if;
end fn_rodzaj_check;
```

- Funkcja sprawdzająca czy w bazie znajduje się już dany rower

```
create or replace function fn_rower_check
(
v_nazwa_roweru in varchar2
)return boolean as v_bool number;
```

```

begin
select count (nazwa_roweru) into v_bool from rower
where nazwa_roweru=v_nazwa_roweru;
if(v_bool=1) then return true;
else return false;
end if;
end fn_rower_check;

```

- Funkcja sprawdzająca czy w bazie znajduje się już dany producent

```

create or replace function fn_producent_check
(
v_nazwa_producenta in varchar2
)return boolean as v_bool number;
begin
select count (nazwa_producenta) into v_bool from producent
where nazwa_producenta=v_nazwa_producenta;
if(v_bool=1) then return true;
else return false;
end if;
end fn_producent_check;

```

- Funkcja sprawdzająca czy w bazie znajduje się już dana nazwa części

```

create or replace function fn_nazwa_czesci_check
(
v_nazwa_czesci in varchar2
)return boolean as v_bool number;
begin
select count (nazwa) into v_bool from nazwa_czesci
where nazwa=v_nazwa_czesci;
if(v_bool=1) then return true;
else return false;

```

```
end if;  
end fn_nazwa_czesci_check;
```

- Funkcja losująca ramę

```
create or replace function fn_rama_random  
return number as v_id_ramy number;  
begin  
select id_ramy into v_id_ramy  
from(select id_ramy from rama  
order by dbms_random.value)  
where rownum=1;  
return v_id_ramy;  
end fn_rama_random;
```

- Funkcja losująca hamulce

```
create or replace function fn_hamulce_random  
return number as v_id_hamulec number;  
begin  
select id_hamulec into v_id_hamulec  
from(select id_hamulec from hamulce  
order by dbms_random.value)  
where rownum=1;  
return v_id_hamulec;  
end fn_hamulce_random;
```

- Funkcja losująca siodełko

```
create or replace function fn_siodelko_random  
return number as v_id_siodelka number;  
begin  
select id_siodelka into v_id_siodelka  
from(select id_siodelka from siodelko
```

```
order by dbms_random.value)
```

```
where rownum=1;
```

```
return v_id_siodelka;
```

```
end fn_siodelko_random;
```

- Funkcja losujaca zestaw kol

```
create or replace function fn_zestaw_kol_random
```

```
return number as v_id_zestawu_kol number;
```

```
begin
```

```
select id_zestawu_kol into v_id_zestawu_kol
```

```
from(select id_zestawu_kol from zestaw_kol
```

```
order by dbms_random.value)
```

```
where rownum=1;
```

```
return v_id_zestawu_kol;
```

```
end fn_zestaw_kol_random;
```

- Funkcja losujaca komplet opon

```
create or replace function fn_komplet_opon_random
```

```
return number as v_id_kompletu_opon number;
```

```
begin
```

```
select id_kompletu_opon into v_id_kompletu_opon
```

```
from(select id_kompletu_opon from komplet_opon
```

```
order by dbms_random.value)
```

```
where rownum=1;
```

```
return v_id_kompletu_opon;
```

```
end fn_komplet_opon_random;
```

- Funkcja losujaca rodzaj roweru

```
create or replace function fn_rodzaj_roweru_random
```

```
return number as v_id_rodzaju number;
```

```
begin
```

```
select id_rodzaju into v_id_rodzaju
```



```

from(select id_rodzaju from rodzaj_roweru
      order by dbms_random.value)
where rownum=1;
return v_id_rodzaju;
end fn_rodzaj_roweru_random;

```

- Funkcja sprawdzająca czy magazyn posiada wystarczającą liczbę rowerów. Zapobiega ona wykonaniu transakcji gdzie klient chce kupić więcej rowerów niż jest na magazynie. Jako parametry należy podać ilość zażyczonych przez klienta rowerów oraz id roweru (w tej kolejności).

```

create or replace function fn_isEnough_rower
(v_potrzeba in number,
 v_id_roweru in number
) return boolean as v_ilosc_na_magazynie number;
begin
select ilosc_na_magazynie into v_ilosc_na_magazynie from rower
where id_roweru=v_id_roweru;
if(v_ilosc_na_magazynie>=v_potrzeba) then return true;
else return false;
end if;
end fn_isEnough_rower;

```

- Funkcja licząca cenę kupowanego roweru (lub kupowanych rowerów) na podstawie części. Jako parametry należy podać (w tej kolejności): id ramy, id zestawu kół, id kompletu opon, id siodełka, id hamulcy oraz ilość rowerów jakie klient zamierza kupić.

```

create or replace function fn_rower_price
(v_id_ramy in number,
 v_id_zestawu_kol in number,
 v_id_kompletu_opon in number,
 v_id_siodełka in number,
 v_id_hamulec in number,
 v_ilosc_rowerow in number

```

```

)return number as

v_cena_ramy number;

v_cena_zestawu_kol number;

v_cena_kompletu_opon number;

v_cena_siodelka number;

v_cena_hamulec number;

v2_ilosc_rowerow number;

v_final number;

begin

select cena_ramy into v_cena_ramy from rama where id_ramy=v_id_ramy;

select cena_zestawu_kol into v_cena_zestawu_kol from zestaw_kol where
id_zestawu_kol=v_id_zestawu_kol;

select cena_kompletu_opon into v_cena_kompletu_opon from komplet_opon where
id_kompletu_opon=v_id_kompletu_opon;

select cena_siodelka into v_cena_siodelka from siodelko where id_siodelka=v_id_siodelka;

select cena_hamulec into v_cena_hamulec from hamulce where id_hamulec=v_id_hamulec;

v_final:=(v_cena_ramy+v_cena_zestawu_kol+v_cena_kompletu_opon+v_cena_siodelka+v_c
ena_hamulec)*v_ilosc_rowerow;

return v_final;

end fn_rower_price;

```

## PROCEDUREY

- Procedura dodająca nazwę części. Parametrem jest nazwa części którą chcemy wprowadzić.

```

create or replace procedure pr_add_nazwa_czesci

(

v_nazwa in varchar2

)as

begin

if(fn_nazwa_czesci_check(v_nazwa)) then dbms_output.put_line ('Taka część już istnieje!');

else insert into nazwa_czesci(nazwa)

values (v_nazwa);

```

```
end if;  
end pr_add_nazwa_czesci;
```

- Procedura dodająca producenta. Parametrem jest nazwa producenta którego chcemy wprowadzić.

```
create or replace procedure pr_add_producent  
(  
  v_nazwa in varchar2  
)as  
begin  
  if(fn_producent_check(v_nazwa)) then dbms_output.put_line ('Ten producent już jest w  
bazie!');  
  
  else insert into producent(nazwa_producenta)  
  values (v_nazwa);  
  
  end if;  
end pr_add_producent;
```

- Procedura dodająca losowy rower. Jako parametry należy podać nową nazwę roweru oraz liczbę rowerów.

```
create or replace procedure pr_add_random_rower  
(  
  v_nazwa_roweru in varchar2,  
  v_ilosc_rowerow in number  
)as  
  v_id_razy number;  
  v_id_zestawu_kol number;  
  v_id_kompletu_opon number;  
  v_id_siodelka number;  
  v_id_hamulec number;  
  v_rodzaj number;  
begin  
  v_id_razy:=fn_razy_random;
```

```

v_id_zestawu_kol:=fn_zestaw_kol_random;
v_id_kompletu_opon:=fn_komplet_opon_random;
v_id_siodelka:=fn_siodelko_random;
v_id_hamulec:=fn_hamulce_random;
v_rodzaj:=fn_rodzaj_roweru_random;

if(fn_rower_check(v_nazwa_roweru))then dbms_output.put_line('Ten rower już istnieje!');
else insert into rower (nazwa_roweru,cena_roweru,ilosc_na_magazynie,
id_razy, id_zestawu_kol,id_kompletu_opon,id_siodelka,id_hamulec, id_rodzaju)
values
(v_nazwa_roweru,fn_rower_price(v_id_razy,v_id_zestawu_kol,v_id_kompletu_opon,v_id_siodelka, v_id_hamulec,v_ilosc_rowerow),
v_ilosc_rowerow,v_id_razy,v_id_zestawu_kol,
v_id_kompletu_opon,v_id_siodelka,v_id_hamulec,v_rodzaj);
end if;

end pr_add_random_rower;

```

- Procedura usuwająca rower. Jako parametr przyjmuje nazwę roweru.

```

create or replace procedure pr_delete_rower
(
v_nazwa_roweru in varchar2
)as
begin
if(fn_rower_check(v_nazwa_roweru))then delete rower where
nazwa_roweru=v_nazwa_roweru;

else dbms_output.put_line('Nie ma takiego roweru!');
end if;

end pr_delete_rower;

```

## WIDOKI

- Najczęściej kupowane rowery

```

create or replace view NAJCZESCIEJ_KUPOWANE_ROWERY as

```

```
select nazwa_roweru as "Model roweru", nazwa_rodzaju as "Rodzaj", sum(ilosc_produktu) as  
"Ilosc sprzedanych rowerow"
```

```
from rower
```

```
join rodzaj_roweru on rower.id_rodzaju = rodzaj_roweru.id_rodzaju
```

```
join faktura on faktura.id_roweru = rower.id_roweru
```

```
group by nazwa_roweru, nazwa_rodzaju;
```

- Zysk z danego modelu roweru

```
create or replace view rowery_zysk as
```

```
select nazwa_roweru as "Model roweru", nazwa_rodzaju as "Rodzaj", cena_roweru as "Cena  
detaliczna", sum(wartosc_faktury) as "Cakowity zarobek"
```

```
from rower
```

```
join rodzaj_roweru on rodzaj_roweru.id_rodzaju = rower.id_rodzaju
```

```
join faktura on faktura.id_roweru = rower.id_roweru
```

```
group by nazwa_roweru, nazwa_rodzaju, cena_roweru;
```

- Wydatki poszczególnych klientów

```
create or replace view wydatki_klientow as
```

```
select nazwisko || ' ' || imie as "Nazwisko i imie", adres, sum(ilosc_produktu) as "Ilosc  
kupionych rowerow",
```

```
sum(wartosc_faktury) as "Wydatek klienta"
```

```
from klient
```

```
join faktura on faktura.id_klienta = klient.id_klienta
```

```
join rower on rower.id_roweru = faktura.id_roweru
```

```
group by nazwisko, imie, adres;
```

- Faktura

```
create or replace view faktura_view as
```

```
select nazwisko || ' ' || imie as "Nazwisko i imie", adres, nazwa_roweru as "Rower",
```

```
cena_roweru as "Cena detaliczna",
```

```
ilosc_produktu as "Ilosc rowerow", wartosc_faktury as "Wartosc", data_sprzedazy as "Data"
```

```
from klient
```

```
join faktura on faktura.id_klienta = klient.id_klienta
```

```
join rower on rower.id_roweru = faktura.id_roweru;
```

- Tygodniowa sprzedaż

```
create or replace view last_week_sales as
```

```
select data_sprzedazy as "Data sprzedaży", count(klient.id_klienta) as "Ilość klientów",
```

```
ilosc_produktu as "Ilość rowerów", nazwa_rodzaju as "Rodzaj"
```

```
from klient
```

```
join faktura on faktura.id_klienta = klient.id_klienta
```

```
join rower on rower.id_roweru = faktura.id_roweru
```

```
join rodzaj_roweru on rodzaj_roweru.id_rodzaju = rower.id_rodzaju
```

```
group by data_sprzedazy, nazwa_rodzaju, ilosc_produktu;
```

## Obsługa

Zalecam skopiowanie kodu z dokumentacji aby uniknąć problemów z wstawianiem danych.

W przypadku gdy chcemy włączyć skrypt z pliku a następnie dodać dane (również z pliku) należy po zakończeniu kompilowania skryptu projekt\_startFINAL.ddl wykonać komendę:

```
Drop trigger tr_update_sold_rower;
```

Następnie wykonać wszystkie inserty z pliku dane.txt i ponownie dodać trigger poleceniem:

```
create or replace trigger tr_update_sold_rower
```

```
before insert on faktura
```

```
for each row
```

```
begin
```

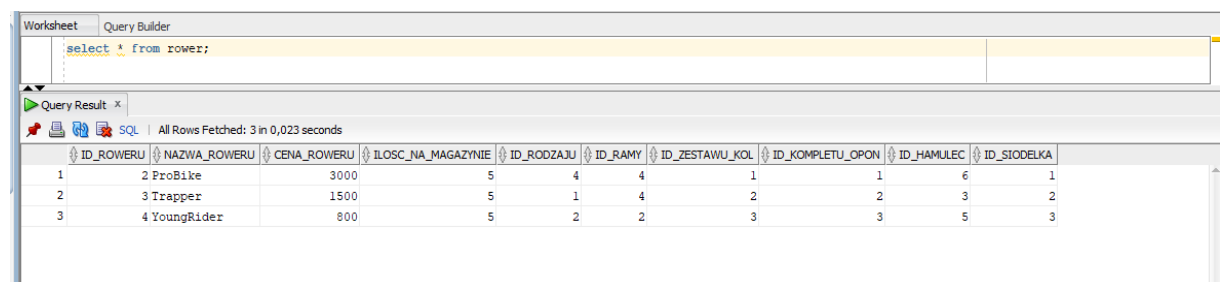
```
update rower
```

```
set ilosc_na_magazynie = ilosc_na_magazynie -:NEW.ilosc_produktu
```

```
where rower.id_roweru = :NEW.id_roweru;
```

```
end tr_update_sold_rower;
```

Po uruchomieniu skryptu instalującego na serwerze w celu sprawdzenia czy podstawowy proces utworzenia struktur i przelania danych przebiegł pomyślnie należy wykonać przykładowo następujące polecenie:



The screenshot shows a database query result in a 'Query Result' window. The query is 'select \* from rower;'. The result is a table with 11 columns and 3 rows of data. The columns are: ID\_ROWERU, NAZWA\_ROWERU, CENA\_ROWERU, ILOSC\_NA\_MAGAZYNIE, ID\_RODZAJU, ID\_RAMY, ID\_ZESTAWU\_KOL, ID\_KOMPLETU\_OPON, ID\_HAMULEC, ID\_SIODELKA, and ID\_KOSZYKA. The rows are: 1, 2 ProBike, 3000, 5, 4, 4, 1, 1, 6, 1, 1; 2, 3 Trapper, 1500, 5, 1, 4, 2, 2, 3, 2, 2; 3, 4 YoungRider, 800, 5, 2, 2, 3, 3, 5, 3, 3.

ID_ROWERU	NAZWA_ROWERU	CENA_ROWERU	ILOSC_NA_MAGAZYNIE	ID_RODZAJU	ID_RAMY	ID_ZESTAWU_KOL	ID_KOMPLETU_OPON	ID_HAMULEC	ID_SIODELKA	ID_KOSZYKA
1	2 ProBike	3000	5	4	4	1	1	6	1	1
2	3 Trapper	1500	5	1	4	2	2	3	2	2
3	4 YoungRider	800	5	2	2	3	3	5	3	3

- Wykonanie procedury

```
begin
pr_add_random_rower('RANDOM',6);
end;
```

Wynik:

select \* from rowery;

Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0,022 seconds

ID_ROWERY	NAZWA_ROWERY	CENA_ROWERY	ILOSC_NA_MAGAZYNIE	ID_RODZAJU	ID_RAMY	ID_ZESTAWU_KOL	ID_KOMPLETU_OPON	ID_HAMULEC	ID_SIODELKA
1	2 ProBike	3000	5	4	4	1	1	6	1
2	3 Trapper	1500	5	1	4	2	2	3	2
3	4 YoungRider	800	5	2	2	3	3	5	3
4	5 RANDOM	13800	6	1	4	2	1	6	1

- Przykłady działania widoków

select \* from rowery\_zysk;

Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0,025 seconds

	Model roweru	Rodzaj	Cena detaliczna	Cakowity zarobek
1	ProBike	Wyczynowy	3000	3500
2	Trapper	Górski	1500	3500

select \* from faktura\_view;

Script Output x Query Result x

SQL | All Rows Fetched: 3 in 0,027 seconds

	Nazwisko i imie	ADRES	Rower	Cena detaliczna	Ilosc rowerow	Wartosc	Data
1	Abacki Adam	ul.Konwaliowa 53	ProBike	3000	1	3500	20/01/01
2	Babacki Bartosz	ul.Sosnowa	Trapper	1500	2	1000	20/01/05
3	Babacki Bartosz	ul.Sosnowa	Trapper	1500	1	2500	20/01/05







Worksheet

Query Builder

```
select * from najczesciej_kupowane_rowery;
```

Script Output x

Query Result x




SQL | All Rows Fetched: 2 in 0,029 seconds

	Model roweru	Rodzaj	Ilosc sprzedanych rowerow
1	Trapper	Górski	3
2	ProBike	Wyczynowy	1

```
select * from last_week_sales;
```

Script Output x

Query Result x

 SQL

All Rows Fetched: 3 in 0,028 seconds

	Data sprzedaży	Ilosc klientow	Ilosc rowerow	Rodzaj
1	20/01/05	1	2	Górski
2	20/01/01	1	1	Wyczynowy
3	20/01/05	1	1	Górski





Worksheet

Query Builder

```
select * from wydatki_klientow;
```

Script Output x

Query Result x

    SQL | All Rows Fetched: 2 in 0,032 seconds

	Nazwisko i imie	ADRES	Ilosc kupionych rowerow	Wydatek klienta
1	Abacki Adam	ul.Konwaliowa 53	1	3500
2	Babacki Bartosz	ul.Sosnowa	3	3500

- Dodawanie do bazy nowego producenta:

worksheet | Query builder

```
begin  
pr_add_producent('NowyProducent');  
end;
```

Script Output x Query Result x

Task completed in 0,073 seconds

PL/SQL procedure successfully completed.

```
select * from producent;
```

Script Output x Query Result x

All Rows Fetched: 7 in 0,032 seconds

	ID_PRODUCENTA	NAZWA_PRODUCENTA
1		1 Shimano
2		2 Shimano
3		3 Cross
4		4 Romet
5		5 BabyKit
6		6 MediumSeries
7		7 NowyProducent

# SKRYPT INSTALUJĄCY:

```
CREATE SEQUENCE seq_id_faktury START WITH 1 INCREMENT BY 1 MAXVALUE 9999999999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_hamulec START WITH 1 INCREMENT BY 1 MAXVALUE 9999999999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_klienta START WITH 1 INCREMENT BY 1 MAXVALUE 9999999999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_kompletu_opon START WITH 1 INCREMENT BY 1 MAXVALUE  
9999999999 NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_nazwy_czesci START WITH 1 INCREMENT BY 1 MAXVALUE 999999999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_producenta START WITH 1 INCREMENT BY 1 MAXVALUE 99999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_ramy START WITH 1 INCREMENT BY 1 MAXVALUE 99999999999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_rodzaju INCREMENT BY 1 NOMAXVALUE NOMINVALUE NOCYCLE  
CACHE 20 NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_roweru START WITH 1 INCREMENT BY 1 MAXVALUE 999999999999999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_siodelka START WITH 1 INCREMENT BY 1 MAXVALUE 99999999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE SEQUENCE seq_id_zestawu_kol START WITH 1 INCREMENT BY 1 MAXVALUE 9999999999  
NOMINVALUE NOCYCLE NOCACHE NOORDER GLOBAL;
```

```
CREATE TABLE faktura (  
    id_klienta    NUMBER(4) NOT NULL,  
    id_roweru     NUMBER(4) NOT NULL,  
    id_faktury    NUMBER(4) NOT NULL,  
    data_sprzedazy DATE NOT NULL,  
    ilosc_produktu NUMBER(2) NOT NULL,  
    wartosc_faktury NUMBER(8, 2) NOT NULL  
)
```

ORGANIZATION HEAP

LOGGING NOCOMPRESS

NOCACHE

NOPARALLEL

NOROWDEPENDENCIES DISABLE ROW MOVEMENT;

ALTER TABLE faktura

ADD CONSTRAINT faktura\_pk PRIMARY KEY ( id\_faktury ) NOT DEFERRABLE ENABLE VALIDATE;

```
CREATE TABLE hamulce (  
    id_hamulec      NUMBER(4) NOT NULL,  
    cena_hamulec     NUMBER(8, 2) NOT NULL,  
    ilosc_hamulec_na_magazynie NUMBER(4) NOT NULL,  
    id_producenta    NUMBER(4) NOT NULL,  
    id_nazwy_czesci   NUMBER(4) NOT NULL  
)
```

ORGANIZATION HEAP

LOGGING NOCOMPRESS

NOCACHE

NOPARALLEL

NOROWDEPENDENCIES DISABLE ROW MOVEMENT;

ALTER TABLE hamulce

```
ADD CONSTRAINT hamulce_pk PRIMARY KEY ( id_hamulec ) NOT DEFERRABLE ENABLE  
VALIDATE;
```

```
CREATE TABLE klient (  
    id_klienta  NUMBER(4) NOT NULL,  
    imie        VARCHAR2(40) NOT NULL,  
    nazwisko    VARCHAR2(40) NOT NULL,  
    adres       VARCHAR2(80) NOT NULL,  
    telefon     NUMBER(15) NOT NULL,  
    email       VARCHAR2(40) NOT NULL  
)
```

```
ORGANIZATION HEAP
```

```
LOGGING NOCOMPRESS
```

```
NOCACHE
```

```
NOPARALLEL
```

```
NOROWDEPENDENCIES DISABLE ROW MOVEMENT;
```

```
ALTER TABLE klient
```

```
ADD CONSTRAINT klient_pk PRIMARY KEY ( id_klienta ) NOT DEFERRABLE ENABLE VALIDATE;
```

```
CREATE TABLE komplet_opon (  
    id_kompletu_opon      NUMBER(4) NOT NULL,  
    cena_kompletu_opon    NUMBER(8, 2) NOT NULL,  
    ilosc_kompletu_opon_na_m  NUMBER(4) NOT NULL,  
    id_producenta         NUMBER(4) NOT NULL,  
    id_nazwy_czesci        NUMBER(4) NOT NULL  
)
```

```
ORGANIZATION HEAP
```

```
LOGGING NOCOMPRESS
```

```
NOCACHE
```

```
NOPARALLEL
```

NOROWDEPENDENCIES DISABLE ROW MOVEMENT;

ALTER TABLE komplet\_opon

ADD CONSTRAINT komplet\_opon\_pk PRIMARY KEY ( id\_kompletu\_opon ) NOT DEFERRABLE  
ENABLE VALIDATE;

CREATE TABLE nazwa\_czesci (

id\_nazwy\_czesci NUMBER(4) NOT NULL,

nazwa VARCHAR2(40) NOT NULL

)

ORGANIZATION HEAP

LOGGING NOCOMPRESS

NOCACHE

NOPARALLEL

NOROWDEPENDENCIES DISABLE ROW MOVEMENT;

ALTER TABLE nazwa\_czesci

ADD CONSTRAINT nazwa\_czesci\_pk PRIMARY KEY ( id\_nazwy\_czesci ) NOT DEFERRABLE  
ENABLE VALIDATE;

CREATE TABLE producent (

id\_producenta NUMBER(4) NOT NULL,

nazwa\_producenta VARCHAR2(40) NOT NULL

)

ORGANIZATION HEAP

LOGGING NOCOMPRESS

NOCACHE

NOPARALLEL

NOROWDEPENDENCIES DISABLE ROW MOVEMENT;

ALTER TABLE producent

```
ADD CONSTRAINT producent_pk PRIMARY KEY ( id_producenta ) NOT DEFERRABLE ENABLE  
VALIDATE;
```

```
CREATE TABLE rama (  
    id_ramy          NUMBER(4) NOT NULL,  
    cena_ramy        NUMBER(8, 2) NOT NULL,  
    ilosc_ramy_na_magazynie NUMBER(4) NOT NULL,  
    id_producenta     NUMBER(4) NOT NULL,  
    id_nazwy_czesci    NUMBER(4) NOT NULL  
)
```

```
ORGANIZATION HEAP
```

```
LOGGING NOCOMPRESS
```

```
NOCACHE
```

```
NOPARALLEL
```

```
NOROWDEPENDENCIES DISABLE ROW MOVEMENT;
```

```
ALTER TABLE rama
```

```
ADD CONSTRAINT rama_pk PRIMARY KEY ( id_ramy ) NOT DEFERRABLE ENABLE VALIDATE;
```

```
CREATE TABLE rodzaj_roweru (  
    id_rodzaju    NUMBER(4) NOT NULL,  
    nazwa_rodzaju VARCHAR2(40) NOT NULL  
)
```

```
ORGANIZATION HEAP
```

```
LOGGING NOCOMPRESS
```

```
NOCACHE
```

```
NOPARALLEL
```

```
NOROWDEPENDENCIES DISABLE ROW MOVEMENT;
```

```
ALTER TABLE rodzaj_roweru
```

```
ADD CONSTRAINT rodzaj_roweru_pk PRIMARY KEY ( id_rodzaju ) NOT DEFERRABLE ENABLE  
VALIDATE;
```

```
CREATE TABLE rower (  
    id_roweru      NUMBER(4) NOT NULL,  
    nazwa_roweru   VARCHAR2(40) NOT NULL,  
    cena_roweru    NUMBER(8, 2) NOT NULL,  
    ilosc_na_magazynie  NUMBER(4) NOT NULL,  
    id_rodzaju     NUMBER(4) NOT NULL,  
    id_ramy        NUMBER(4) NOT NULL,  
    id_zestawu_kol  NUMBER(4) NOT NULL,  
    id_kompletu_opon  NUMBER(4) NOT NULL,  
    id_hamulec     NUMBER(4) NOT NULL,  
    id_siodelka    NUMBER(4) NOT NULL  
)
```

ORGANIZATION HEAP

LOGGING NOCOMPRESS

NOCACHE

NOPARALLEL

NOROWDEPENDENCIES DISABLE ROW MOVEMENT;

ALTER TABLE rower

ADD CONSTRAINT rower\_pk PRIMARY KEY ( id\_roweru ) NOT DEFERRABLE ENABLE VALIDATE;

```
CREATE TABLE siodelko (  
    id_siodelka    NUMBER(4) NOT NULL,  
    cena_siodelka  NUMBER(8, 2) NOT NULL,  
    ilosc_siodelek_na_magazynie  NUMBER(4) NOT NULL,  
    id_producenta  NUMBER(4) NOT NULL,  
    id_nazwy_czesci  NUMBER(4) NOT NULL  
)
```

ORGANIZATION HEAP

LOGGING NOCOMPRESS



NOCACHE

NOPARALLEL

NOROWDEPENDENCIES DISABLE ROW MOVEMENT;

ALTER TABLE siodelko

ADD CONSTRAINT siodelko\_pk PRIMARY KEY ( id\_siodelka ) NOT DEFERRABLE ENABLE  
VALIDATE;

CREATE TABLE zestaw\_kol (

id\_zestawu\_kol        NUMBER(4) NOT NULL,

cena\_zestawu\_kol     NUMBER(8, 2) NOT NULL,

ilosc\_zestawow\_kol\_na\_m   NUMBER(4) NOT NULL,

id\_producenta        NUMBER(4) NOT NULL,

id\_nazwy\_czesci       NUMBER(4) NOT NULL

)

ORGANIZATION HEAP

LOGGING NOCOMPRESS

NOCACHE

NOPARALLEL

NOROWDEPENDENCIES DISABLE ROW MOVEMENT;

ALTER TABLE zestaw\_kol

ADD CONSTRAINT zestaw\_kol\_pk PRIMARY KEY ( id\_zestawu\_kol ) NOT DEFERRABLE ENABLE  
VALIDATE;

ALTER TABLE rower

ADD CONSTRAINT hamulce\_fk FOREIGN KEY ( id\_hamulec )

REFERENCES hamulce ( id\_hamulec )

NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE faktura

ADD CONSTRAINT klient\_fk FOREIGN KEY ( id\_klienta )

```
REFERENCES klient ( id_klienta )  
NOT DEFERRABLE ENABLE VALIDATE;
```

ALTER TABLE rower

```
ADD CONSTRAINT komplet_opon_fk FOREIGN KEY ( id_kompletu_opon )  
REFERENCES komplet_opon ( id_kompletu_opon )  
NOT DEFERRABLE ENABLE VALIDATE;
```

ALTER TABLE rama

```
ADD CONSTRAINT nazwa_czesci_fk FOREIGN KEY ( id_nazwy_czesci )  
REFERENCES nazwa_czesci ( id_nazwy_czesci )  
NOT DEFERRABLE ENABLE VALIDATE;
```

ALTER TABLE zestaw\_kol

```
ADD CONSTRAINT nazwa_czesci_fkv2 FOREIGN KEY ( id_nazwy_czesci )  
REFERENCES nazwa_czesci ( id_nazwy_czesci )  
NOT DEFERRABLE ENABLE VALIDATE;
```

ALTER TABLE komplet\_opon

```
ADD CONSTRAINT nazwa_czesci_fkv3 FOREIGN KEY ( id_nazwy_czesci )  
REFERENCES nazwa_czesci ( id_nazwy_czesci )  
NOT DEFERRABLE ENABLE VALIDATE;
```

ALTER TABLE hamulce

```
ADD CONSTRAINT nazwa_czesci_fkv4 FOREIGN KEY ( id_nazwy_czesci )  
REFERENCES nazwa_czesci ( id_nazwy_czesci )  
NOT DEFERRABLE ENABLE VALIDATE;
```

ALTER TABLE siodelko

```
ADD CONSTRAINT nazwa_czesci_fkv5 FOREIGN KEY ( id_nazwy_czesci )  
REFERENCES nazwa_czesci ( id_nazwy_czesci )
```

NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE zestaw\_kol

ADD CONSTRAINT producent\_fk FOREIGN KEY ( id\_producenta )

REFERENCES producent ( id\_producenta )

NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE komplet\_opon

ADD CONSTRAINT producent\_fkv2 FOREIGN KEY ( id\_producenta )

REFERENCES producent ( id\_producenta )

NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE hamulce

ADD CONSTRAINT producent\_fkv3 FOREIGN KEY ( id\_producenta )

REFERENCES producent ( id\_producenta )

NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE siodelko

ADD CONSTRAINT producent\_fkv4 FOREIGN KEY ( id\_producenta )

REFERENCES producent ( id\_producenta )

NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE rama

ADD CONSTRAINT producent\_fkv5 FOREIGN KEY ( id\_producenta )

REFERENCES producent ( id\_producenta )

NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE rower

ADD CONSTRAINT rama\_fk FOREIGN KEY ( id\_ramy )

REFERENCES rama ( id\_ramy )

NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE rower

ADD CONSTRAINT rodzaj\_roweru\_fk FOREIGN KEY ( id\_rodzaju )  
REFERENCES rodzaj\_roweru ( id\_rodzaju )  
NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE faktura

ADD CONSTRAINT rower\_fk FOREIGN KEY ( id\_roweru )  
REFERENCES rower ( id\_roweru )  
NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE rower

ADD CONSTRAINT siodelko\_fk FOREIGN KEY ( id\_siodelka )  
REFERENCES siodelko ( id\_siodelka )  
NOT DEFERRABLE ENABLE VALIDATE;

ALTER TABLE rower

ADD CONSTRAINT zestaw\_kol\_fk FOREIGN KEY ( id\_zestawu\_kol )  
REFERENCES zestaw\_kol ( id\_zestawu\_kol )  
NOT DEFERRABLE ENABLE VALIDATE;

CREATE OR REPLACE TRIGGER te\_id\_kompletu\_opon

BEFORE INSERT ON komplet\_opon

FOR EACH ROW

begin

:NEW.id\_kompletu\_opon:=seq\_id\_kompletu\_opon.nextval;

end;

/

```
CREATE OR REPLACE TRIGGER tr_id_faktury
    BEFORE INSERT ON faktura
    FOR EACH ROW
begin
:NEW.id_faktury:=seq_id_faktury.nextval;
end;
/
```

```
CREATE OR REPLACE TRIGGER tr_id_hamulec
    BEFORE INSERT ON hamulce
    FOR EACH ROW
begin
:NEW.id_hamulec:=seq_id_hamulec.nextval;
end;
/
```

```
CREATE OR REPLACE TRIGGER tr_id_klienta
    BEFORE INSERT ON klient
    FOR EACH ROW
begin
:NEW.id_klienta:=seq_id_klienta.nextval;
end;
/
```

```
CREATE OR REPLACE TRIGGER tr_id_nazwy_czesci
    BEFORE INSERT ON nazwa_czesci
    FOR EACH ROW
begin
:NEW.id_nazwy_czesci:=seq_id_nazwy_czesci.nextval;
end;
/
```

```
CREATE OR REPLACE TRIGGER tr_id_producenta
    BEFORE INSERT ON producent
    FOR EACH ROW
begin
:NEW.id_producenta:=seq_id_producenta.nextval;
end;
/
```

```
CREATE OR REPLACE TRIGGER tr_id_ramy
    BEFORE INSERT ON rama
    FOR EACH ROW
begin
:NEW.id_ramy:=seq_id_ramy.nextval;
end;
/
```

```
CREATE OR REPLACE TRIGGER tr_id_rodzaju
    BEFORE INSERT ON rodzaj_roweru
    FOR EACH ROW
begin
:NEW.id_rodzaju:=seq_id_rodzaju.nextval;
end;
/
```

```
CREATE OR REPLACE TRIGGER tr_id_roweru
    BEFORE INSERT ON rower
    FOR EACH ROW
begin
:NEW.id_roweru:=seq_id_roweru.nextval;
end;
```

/

```
CREATE OR REPLACE TRIGGER tr_id_siodelka
```

```
    BEFORE INSERT ON siodelko
```

```
    FOR EACH ROW
```

```
begin
```

```
:NEW.id_siodelka:=seq_id_siodelka.nextval;
```

```
end;
```

/

```
CREATE OR REPLACE TRIGGER tr_id_zestawu_kol
```

```
    BEFORE INSERT ON zestaw_kol
```

```
    FOR EACH ROW
```

```
begin
```

```
:NEW.id_zestawu_kol:=seq_id_zestawu_kol.nextval;
```

```
end;
```

/

```
create or replace function fn_rodzaj_check
```

```
(
```

```
v_nazwa_rodzaju in varchar2
```

```
)return boolean as v_bool number;
```

```
begin
```

```
select count (nazwa_rodzaju) into v_bool from rodzaj_roweru
```

```
where nazwa_rodzaju=v_nazwa_rodzaju;
```

```
if(v_bool=1) then return true;
```

```
else return false;
```

```
end if;
```

```
end fn_rodzaj_check;
```

/

```
create or replace function fn_rower_check
(
v_nazwa_roweru in varchar2
)return boolean as v_bool number;
begin
select count (nazwa_roweru) into v_bool from rower
where nazwa_roweru=v_nazwa_roweru;
if(v_bool=1) then return true;
else return false;
end if;
end fn_rower_check;
/
```

```
create or replace function fn_producent_check
(
v_nazwa_producenta in varchar2
)return boolean as v_bool number;
begin
select count (nazwa_producenta) into v_bool from producent
where nazwa_producenta=v_nazwa_producenta;
if(v_bool=1) then return true;
else return false;
end if;
end fn_producent_check;
/
```

```
create or replace function fn_nazwa_czesci_check
(
v_nazwa_czesci in varchar2
)return boolean as v_bool number;
begin
select count (nazwa) into v_bool from nazwa_czesci
```



```

where nazwa=v_nazwa_czesci;
if(v_bool=1) then return true;
else return false;
end if;
end fn_nazwa_czesci_check;
/

create or replace function fn_rama_random
return number as v_id_ramy number;
begin
select id_ramy into v_id_ramy
from(select id_ramy from rama
      order by dbms_random.value)
where rownum=1;
return v_id_ramy;
end fn_rama_random;
/

create or replace function fn_hamulce_random
return number as v_id_hamulec number;
begin
select id_hamulec into v_id_hamulec
from(select id_hamulec from hamulce
      order by dbms_random.value)
where rownum=1;
return v_id_hamulec;
end fn_hamulce_random;
/

create or replace function fn_siodelko_random
return number as v_id_siodelka number;
begin
select id_siodelka into v_id_siodelka
from(select id_siodelka from siodelko

```

```

        order by dbms_random.value)
where rownum=1;
return v_id_siodelka;
end fn_siodelko_random;

/

create or replace function fn_zestaw_kol_random
return number as v_id_zestawu_kol number;
begin
select id_zestawu_kol into v_id_zestawu_kol
from(select id_zestawu_kol from zestaw_kol
      order by dbms_random.value)
where rownum=1;
return v_id_zestawu_kol;
end fn_zestaw_kol_random;

/

create or replace function fn_komplet_opon_random
return number as v_id_kompletu_opon number;
begin
select id_kompletu_opon into v_id_kompletu_opon
from(select id_kompletu_opon from komplet_opon
      order by dbms_random.value)
where rownum=1;
return v_id_kompletu_opon;
end fn_komplet_opon_random;

/

create or replace function fn_rodzaj_roweru_random
return number as v_id_rodzaju number;
begin
select id_rodzaju into v_id_rodzaju
from(select id_rodzaju from rodzaj_roweru
      order by dbms_random.value)

```

```

where rownum=1;
return v_id_rodzaju;
end fn_rodzaj_roweru_random;
/

create or replace function fn_isEnough_rower
(v_potrzeba in number,
v_id_roweru in number
)return boolean as v_ilosc_na_magazynie number;
begin
select ilosc_na_magazynie into v_ilosc_na_magazynie from rower
where id_roweru=v_id_roweru;
if(v_ilosc_na_magazynie>=v_potrzeba) then return true;
else return false;
end if;
end fn_isEnough_rower;
/

create or replace procedure pr_add_nazwa_czesci
(
v_nazwa in varchar2
)as
begin
if(fn_nazwa_czesci_check(v_nazwa)) then dbms_output.put_line ('Taka część już istnieje!');
else insert into nazwa_czesci(nazwa)
values (v_nazwa);
end if;
end pr_add_nazwa_czesci;
/

create or replace procedure pr_add_producent
(
v_nazwa in varchar2
)as

```

```

begin
if(fn_producent_check(v_nazwa)) then dbms_output.put_line ('Ten producent już jest w bazie!');
else insert into producent(nazwa_producenta)
values (v_nazwa);
end if;
end pr_add_producent;
/
create or replace procedure pr_add_random_rower
(
v_nazwa_roweru in varchar2,
v_ilosc_rowerow in number
)as
v_id_ramy number;
v_id_zestawu_kol number;
v_id_kompletu_opon number;
v_id_siodelka number;
v_id_hamulec number;
v_rodzaj number;
begin
v_id_ramy:=fn_rama_random;
v_id_zestawu_kol:=fn_zestaw_kol_random;
v_id_kompletu_opon:=fn_komplet_opon_random;
v_id_siodelka:=fn_siodelko_random;
v_id_hamulec:=fn_hamulce_random;
v_rodzaj:=fn_rodzaj_roweru_random;
if(fn_rower_check(v_nazwa_roweru))then dbms_output.put_line('Ten rower już istnieje!');
else insert into rower (nazwa_roweru,cena_roweru,ilosc_na_magazynie,
id_ramy, id_zestawu_kol,id_kompletu_opon,id_siodelka,id_hamulec, id_rodzaju)
values
(v_nazwa_roweru,fn_rower_price(v_id_ramy,v_id_zestawu_kol,v_id_kompletu_opon,v_id_siode
lka, v_id_hamulec,v_ilosc_rowerow),

```

```

v_ilosc_rowerow,v_id_ramy,v_id_zestawu_kol,
v_id_kompletu_opon,v_id_siodelka,v_id_hamulec,v_rodzaj));

end if;

end pr_add_random_rower;

/

create or replace procedure pr_delete_rower
(
v_nazwa_roweru in varchar2
)as
begin
if(fn_rower_check(v_nazwa_roweru))then delete rower where nazwa_roweru=v_nazwa_roweru;
else dbms_output.put_line('Nie ma takiego roweru!');
end if;

end pr_delete_rower;

/

create or replace view NAJCZESCIEJ_KUPOWANE_ROWERY as

select nazwa_roweru as "Model roweru", nazwa_rodzaju as "Rodzaj", sum(ilosc_produktu) as
"Ilosc sprzedanych rowerow"

from rower

join rodzaj_roweru on rower.id_rodzaju = rodzaj_roweru.id_rodzaju

join faktura on faktura.id_roweru = rower.id_roweru

group by nazwa_roweru, nazwa_rodzaju;

/

create or replace view rowery_zysk as

select nazwa_roweru as "Model roweru", nazwa_rodzaju as "Rodzaj",cena_roweru as "Cena
detaliczna", sum(wartosc_faktury) as "Cakowity zarobek"

from rower

join rodzaj_roweru on rodzaj_roweru.id_rodzaju = rower.id_rodzaju

join faktura on faktura.id_roweru = rower.id_roweru

group by nazwa_roweru, nazwa_rodzaju, cena_roweru;

/

```

create or replace view wydatki\_klientow as

select nazwisko || ' ' || imie as "Nazwisko i imie", adres, sum(ilosc\_produktu) as "Ilosc kupionych  
rowerow",

sum(wartosc\_faktury) as "Wydatek klienta"

from klient

join faktura on faktura.id\_klienta = klient.id\_klienta

join rower on rower.id\_roweru = faktura.id\_roweru

group by nazwisko, imie, adres;

/

create or replace view faktura\_view as

select nazwisko || ' ' || imie as "Nazwisko i imie", adres, nazwa\_roweru as "Rower",

cena\_roweru as "Cena detaliczna",

ilosc\_produktu as "Ilosc rowerow", wartosc\_faktury as "Wartosc", data\_sprzedazy as "Data"

from klient

join faktura on faktura.id\_klienta = klient.id\_klienta

join rower on rower.id\_roweru = faktura.id\_roweru;

/

create or replace view last\_week\_sales as

select data\_sprzedazy as "Data sprzedazy", count(klient.id\_klienta) as "Ilosc klientow",

ilosc\_produktu as "Ilosc rowerow", nazwa\_rodzaju as "Rodzaj"

from klient

join faktura on faktura.id\_klienta = klient.id\_klienta

join rower on rower.id\_roweru = faktura.id\_roweru

join rodzaj\_roweru on rodzaj\_roweru.id\_rodzaju = rower.id\_rodzaju

group by data\_sprzedazy, nazwa\_rodzaju, ilosc\_produktu;

/

-----  
-----

insert into rodzaj\_roweru(nazwa\_rodzaju)

values ('Górski');

```
insert into rodzaj_roweru(nazwa_rodzaju)
values ('Dzieciecy');
insert into rodzaj_roweru(nazwa_rodzaju)
values ('Szosowy');
insert into rodzaj_roweru(nazwa_rodzaju)
values ('Wyczynowy');
insert into rodzaj_roweru(nazwa_rodzaju)
values ('BMX');
```

```
insert into Producent(nazwa_producenta)
values ('Cross');
insert into Producent(nazwa_producenta)
values ('Shimano');
insert into Producent(nazwa_producenta)
values ('Romet');
```

```
insert into nazwa_czesci(nazwa)
values ('SteelSeries');
insert into nazwa_czesci(nazwa)
values ('MediumSeries');
insert into nazwa_czesci(nazwa)
values ('UltimateSeries');
```

```
insert into rama(cena_ramy,ilosc_ramy_na_magazynie,id_producenta,id_nazwy_czesci)
values (600,10,1,1);
insert into rama(cena_ramy,ilosc_ramy_na_magazynie,id_producenta,id_nazwy_czesci)
values (100,10,1,2);
insert into rama(cena_ramy,ilosc_ramy_na_magazynie,id_producenta,id_nazwy_czesci)
values (300,15,2,3);
insert into rama(cena_ramy,ilosc_ramy_na_magazynie,id_producenta,id_nazwy_czesci)
values (900,2,2,1);
```

```
insert into  
hamulce(cena_hamulec,ilosc_hamulec_na_magazynie,id_producenta,id_nazwy_czesci)  
  
values (600,10,1,1);
```

```
insert into  
hamulce(cena_hamulec,ilosc_hamulec_na_magazynie,id_producenta,id_nazwy_czesci)  
  
values (50,10,1,2);
```

```
insert into  
hamulce(cena_hamulec,ilosc_hamulec_na_magazynie,id_producenta,id_nazwy_czesci)  
  
values (600,10,1,3);
```

```
insert into siodelko(cena_siodelka,ilosc_siodelek_na_magazynie,id_producenta,id_nazwy_czesci)  
  
values (300,10,1,3);
```

```
insert into siodelko(cena_siodelka,ilosc_siodelek_na_magazynie,id_producenta,id_nazwy_czesci)  
  
values (200,10,1,1);
```

```
insert into siodelko(cena_siodelka,ilosc_siodelek_na_magazynie,id_producenta,id_nazwy_czesci)  
  
values (100,10,2,3);
```

```
insert into  
zestaw_kol(cena_zestawu_kol,ilosc_zestawow_kol_na_m,id_producenta,id_nazwy_czesci)  
  
values (300,10,1,3);
```

```
insert into  
zestaw_kol(cena_zestawu_kol,ilosc_zestawow_kol_na_m,id_producenta,id_nazwy_czesci)  
  
values (200,10,1,2);
```

```
insert into  
zestaw_kol(cena_zestawu_kol,ilosc_zestawow_kol_na_m,id_producenta,id_nazwy_czesci)  
  
values (100,10,3,3);
```

```
insert into  
komplet_opon(cena_kompletu_opon,ilosc_kompletu_opon_na_m,id_producenta,id_nazwy_czesci)  
  
values (300,10,1,3);
```

```
insert into  
komplet_opon(cena_kompletu_opon,ilosc_kompletu_opon_na_m,id_producenta,id_nazwy_czesci)  
  
values (300,10,1,3);
```



```
values (200,10,2,3);
```

```
insert into
```

```
komplet_opon(cena_kompletu_opon,ilosc_kompletu_opon_na_m,id_producenta,id_nazwy_czes  
ci)
```

```
values (100,10,1,3);
```

```
insert into rower(nazwa_roweru,cena_roweru,ilosc_na_magazynie, id_rodzaju, id_ramy,  
id_zestawu_kol,id_kompletu_opon,id_hamulec,id_siodelka)
```

```
values ('ProBike',3000,5,4,4,1,1,1,1);
```

```
insert into rower(nazwa_roweru,cena_roweru,ilosc_na_magazynie, id_rodzaju, id_ramy,  
id_zestawu_kol,id_kompletu_opon,id_hamulec,id_siodelka)
```

```
values ('Trapper',1500,5,1,4,2,2,3,2);
```

```
insert into rower(nazwa_roweru,cena_roweru,ilosc_na_magazynie, id_rodzaju, id_ramy,  
id_zestawu_kol,id_kompletu_opon,id_hamulec,id_siodelka)
```

```
values ('YoungRider',800,1,2,2,3,3,1,3);
```

```
insert into klient (imie,nazwisko,adres, telefon,email)
```

```
VALUES ('Adam', 'Abacki', 'ul.Konwaliowa 53',123456789, 'abackiadam@poczta.pl');
```

```
insert into klient (imie,nazwisko,adres, telefon,email)
```

```
VALUES ('Bartosz', 'Babacki', 'ul.Sosnowa',987654321, 'babackibartosz@poczta.pl');
```

```
insert into faktura(id_klienta,id_roweru,data_sprzedazy,
```

```
ilosc_produktu,wartosc_faktury)
```

```
values(1,2,'20/01/01',1,3500);
```

```
insert into faktura(id_klienta,id_roweru,data_sprzedazy,
```

```
ilosc_produktu,wartosc_faktury)
```

```
values(2,2,'20/01/05',1,2500);
```

```
insert into faktura(id_klienta,id_roweru,data_sprzedazy,
```

```
ilosc_produktu,wartosc_faktury)
```

```
values(2,1,'20/01/05',2,1000);
```

```

create or replace trigger tr_update_sold_rower
before insert on faktura
for each row
begin
update rower
set ilosc_na_magazynie = ilosc_na_magazynie -:NEW.ilosc_produktu
where rower.id_roweru = :NEW.id_roweru;
end tr_update_sold_rower;
/

create or replace trigger tr_update_sold_rama
before insert on rower
for each row
begin
update rama
set ilosc_ramy_na_magazynie = ilosc_ramy_na_magazynie -:NEW.ilosc_na_magazynie
where rama.id_ramy = :NEW.id_ramy;
end tr_update_sold_rama;
/

create or replace trigger tr_update_sold_hamulec
before insert on rower
for each row
begin
update hamulce
set ilosc_hamulec_na_magazynie = ilosc_hamulec_na_magazynie -:NEW.ilosc_na_magazynie
where hamulce.id_hamulec = :NEW.id_hamulec;
end tr_update_sold_hamulec;
/

```

```

create or replace trigger tr_update_sold_siodelko
before insert on rowser
for each row
begin
update siodelko
set ilosc_siodelek_na_magazynie = ilosc_siodelek_na_magazynie -:NEW.ilosc_na_magazynie
where siodelko.id_siodelka = :NEW.id_siodelka;
end tr_update_sold_siodelko;
/

create or replace trigger tr_update_sold_zestaw_kol
before insert on rowser
for each row
begin
update zestaw_kol
set ilosc_zestawow_kol_na_m = ilosc_zestawow_kol_na_m -:NEW.ilosc_na_magazynie
where zestaw_kol.id_zestawu_kol = :NEW.id_zestawu_kol;
end tr_update_sold_zestaw_kol;
/

create or replace trigger tr_update_sold_komplet_opon
before insert on rowser
for each row
begin
update komplet_opon
set ilosc_kompletu_opon_na_m = ilosc_kompletu_opon_na_m -:NEW.ilosc_na_magazynie
where komplet_opon.id_kompletu_opon = :NEW.id_kompletu_opon;
end tr_update_sold_komplet_opon;

```

# Skrypt deinstalujący:

**DROP TABLE faktura CASCADE CONSTRAINTS;**

**DROP TABLE hamulce CASCADE CONSTRAINTS;**

**DROP TABLE klient CASCADE CONSTRAINTS;**

**DROP TABLE komplet\_opon CASCADE CONSTRAINTS;**

**DROP TABLE nazwa\_czesci CASCADE CONSTRAINTS;**

**DROP TABLE producent CASCADE CONSTRAINTS;**

**DROP TABLE rama CASCADE CONSTRAINTS;**

**DROP TABLE rodzaj\_roweru CASCADE CONSTRAINTS;**

**DROP TABLE rower CASCADE CONSTRAINTS;**

**DROP TABLE siodelko CASCADE CONSTRAINTS;**

**DROP TABLE zestaw\_kol CASCADE CONSTRAINTS;**

**DROP SEQUENCE seq\_id\_faktury;**

**DROP SEQUENCE seq\_id\_hamulec;**

**DROP SEQUENCE seq\_id\_klienta;**

**DROP SEQUENCE seq\_id\_kompletu\_opon;**

**DROP SEQUENCE seq\_id\_nazwy\_czesci;**

**DROP SEQUENCE seq\_id\_producenta;**

**DROP SEQUENCE seq\_id\_ramy;**

**DROP SEQUENCE seq\_id\_rodzaju;**

**DROP SEQUENCE seq\_id\_roweru;**

**DROP SEQUENCE seq\_id\_siodelka;**

**DROP SEQUENCE seq\_id\_zestawu\_kol;**

**DROP FUNCTION FN\_RODZAJ\_CHECK;**

**DROP function fn\_rower\_check;**

**DROP function fn\_producent\_check;**

**DROP function fn\_nazwa\_czesci\_check;**

**DROP function fn\_rama\_random;**

**DROP function fn\_hamulce\_random;**

**DROP function fn\_siodelko\_random;**

**DROP function fn\_zestaw\_kol\_random;**

```
DROP function fn_komplet_opon_random;  
DROP function fn_rodzaj_roweru_random;  
DROP function fn_isEnough_rower;  
DROP procedure pr_add_nazwa_czesci;  
DROP procedure pr_add_producent;  
DROP procedure pr_add_random_rower;  
DROP procedure pr_delete_rower;  
DROP view NAJCZESCIEJ_KUPOWANE_ROWERY;  
DROP view rowery_zysk;  
DROP view wydatki_klientow;  
DROP view faktura_view;  
DROP view last_week_sales;
```