Database Management System 2 Endterm

**Zhumadulla Adilet**

**Tanatar Diyat**

**Kudaibergenov Elzhas**

**Aitbay Damir**

**Orynbasar Sabyrzhan**

1. Procedure which does group by information

CREATE OR REPLACE PROCEDURE group\_info AS

CURSOR c IS

SELECT Year, COUNT(\*) AS num

FROM Product\_company

WHERE Year > 2000 --> gets data greater than 2000 years

GROUP BY Year;

BEGIN

FOR r IN c LOOP

DBMS\_OUTPUT.PUT\_LINE(r.Year || ': ' || r.num);

END LOOP;

END;

--> Calling the procedure which is groups car years

BEGIN

group\_info;

END;

1. Function which counts the number of records

CREATE OR REPLACE FUNCTION num\_of\_rec(Employees VARCHAR2) RETURN NUMBER AS

num\_of\_rec NUMBER;

BEGIN

EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' || Employees INTO num\_of\_rec;

return num\_of\_rec;

END;

-- Calling the function

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Number of records :' || num\_of\_rec('Employees'));

END;

1. Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters

DECLARE

p\_mark Product\_company.Mark%type :=:Enter;

p\_model Product\_company.Model%type;

p\_price Product\_company.Price%type;

mark\_length NUMBER;

invalid\_mark\_name EXCEPTION;

BEGIN

mark\_length := LENGTH(p\_mark);

IF mark\_length < 5 THEN

RAISE invalid\_mark\_name;

ELSE

SELECT Mark, Model, Price INTO p\_mark, p\_model, p\_price

FROM Product\_company

WHERE Mark = p\_mark;

DBMS\_OUTPUT.PUT\_LINE('Mark: ' || p\_mark);

DBMS\_OUTPUT.PUT\_LINE('Model: ' || p\_model);

DBMS\_OUTPUT.PUT\_LINE('Price: ' || p\_price);

END IF;

EXCEPTION

WHEN invalid\_mark\_name THEN

dbms\_output.put\_line('Mark must be at least 5 characters long');

WHEN no\_data\_found THEN

dbms\_output.put\_line('No such mark');

WHEN others THEN

dbms\_output.put\_line('Sorry, error!');

END;

1. Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

CREATE OR REPLACE PROCEDURE row\_count AS

update\_row NUMBER;

BEGIN

UPDATE product\_company

SET price = price || ' -50% discount'

WHERE year = '2009';

update\_row := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Number of rows affected: ' || update\_row);

END;

BEGIN

row\_count;

END;

1. Create a trigger before insert on any entity which will show the current number of rows in the table

CREATE OR REPLACE TRIGGER count\_trigger

BEFORE INSERT ON payment

DECLARE

count\_pay NUMBER;

BEGIN

SELECT COUNT(DISTINCT payment\_type) INTO count\_pay FROM payment;

DBMS\_OUTPUT.PUT\_LINE('How many kinds of payment methods we have: ' || count\_pay);

FOR i IN (SELECT DISTINCT payment\_type FROM payment) LOOP

DBMS\_OUTPUT.PUT\_LINE('Payment method name: ' || i.payment\_type);

END LOOP;

END;

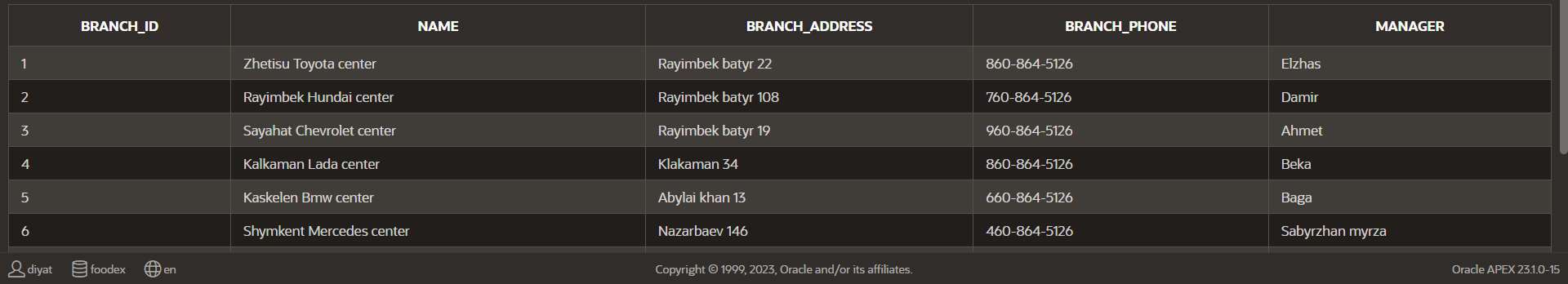
INSERT INTO payment

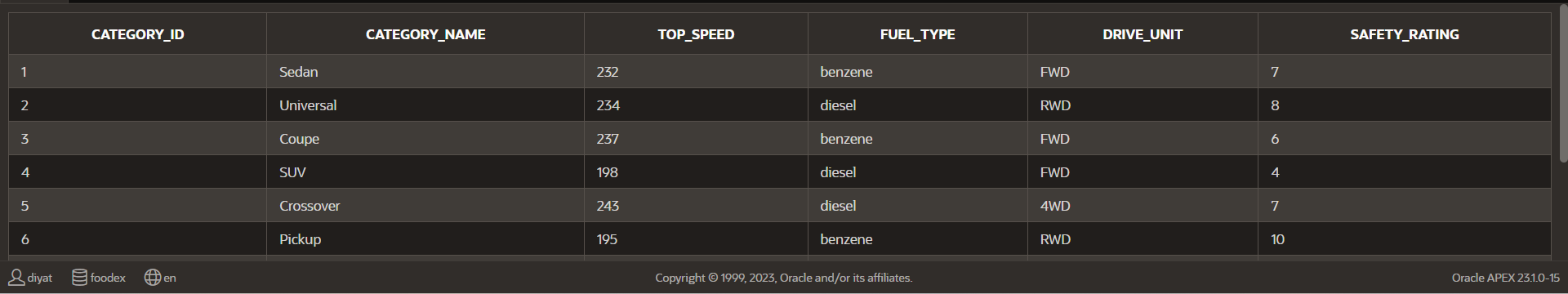
VALUES (102, 102, 102, 'cash', 'cash', '450000');

**Normalization in DBMS - 1NF, 2NF, 3NF, 4NF**

## **1NF (First Normal Form) Rules**

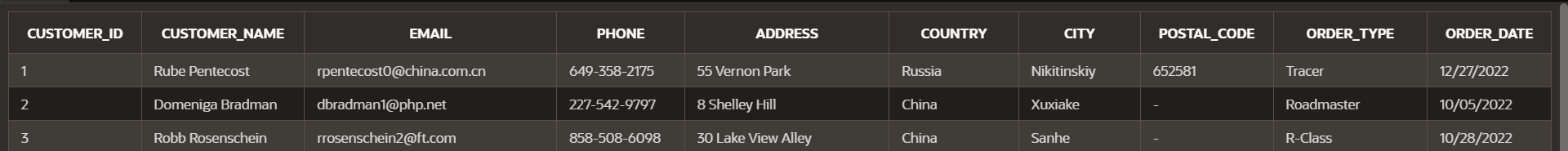
* Each table cell should contain a single value
* Each record needs to be unique

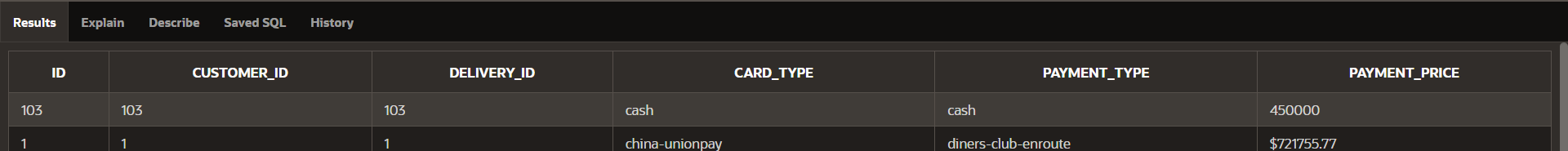




**2NF (Second Normal Form) Rules**

* Be in 1NF
* Single column primary key that does not functionally dependant on any subset of candidate key relation





**3NF (Third Normal Form) Rules**

* Be in 2NF----->1NF + 2NF
* Has no transitive functional dependencies

