Task 1

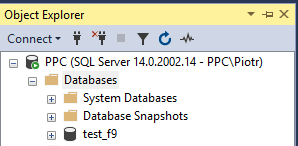
1. Choose New Query option for opening SQL worksheet window.



1. Define new database named test\_yourname using CREATE DATABASE statement.

CREATE DATABASE test\_f9;

1. Refresh Object Explorer panel to see your new database.



1. Check the name of the database you are connected to. You can change a current database using the statement: USE database\_name

USE test\_f9;

1. Define table named BANDS, which consists of the following columns: band\_id – INTEGER, primery key, name – VARCHAR limited to 40 CHARacters, origin\_country - VARCHAR limited to 50 CHARacters, formed\_year – INTEGER.

CREATE TABLE BANDS (

band\_id INTEGER PRIMARY KEY,

name VARCHAR(40),

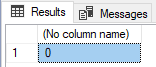
origin\_country VARCHAR(50),

formed\_year INTEGER

);

1. Check the number of records in that table using SELECT count(\*) … statement.

SELECT COUNT(\*) FROM BANDS;



1. Insert into the table one record: name: The Beatles, origin\_country: England, formed\_year 1960

INSERT INTO BANDS (band\_id, name, origin\_country, formed\_year)

VALUES (1, 'The Beatles', 'England', 1960);

1. Display all the data using SELECT statement.

SELECT \* FROM BANDS;



1. Check the number of records in that table again.

SELECT COUNT(\*) FROM BANDS;



1. Create another table named MEMBERS consisted of: memeber\_id - INTEGER incremental from 100 by 1, band\_id - int, surname - VARCHAR limited to 60 CHARacters, name VARCHAR limited to 50 CHARacters.

CREATE TABLE MEMBERS (

member\_id INTEGER PRIMARY KEY IDENTITY(100, 1),

band\_id INTEGER,

surname VARCHAR(60),

name VARCHAR(50),

);

1. Add foreign key on band\_id column of MEMBERS table, which references BANDS table.

ALTER TABLE MEMBERS ADD

CONSTRAINT fk\_members\_bands FOREIGN KEY (band\_id) REFERENCES BANDS(band\_id);

1. Insert into that table 2 records for The Beatles band: John Lennon and Paul McCartney.

DECLARE @band INT;

SELECT @band = band\_id

FROM BANDS

WHERE name = 'The Beatles';

INSERT INTO MEMBERS (band\_id, surname, name)

VALUES (@band, 'Lennon', 'John');

INSERT INTO MEMBERS (band\_id, surname, name)

VALUES (@band, 'McCartney', 'Paul');

1. Insert into BANDS table another record: name: Queen, origin\_country: Great Britain, formed\_year: 1971

INSERT INTO BANDS (band\_id, name, origin\_country, formed\_year)

VALUES (2, 'Queen', 'Great Britain', 1971);

1. Insert another member: Freddie Mercury.

DECLARE @band INT;

SELECT @band = band\_id

FROM BANDS

WHERE name = 'Queen';

INSERT INTO MEMBERS (band\_id, surname, name)

VALUES (@band, 'Mercury', 'Freddie');

1. Add constraint, which doesn’t allow entering year earlier than 1920.

ALTER TABLE BANDS ADD CHECK (formed\_year >= 1920);

1. Add another record to ensure that the constraint works properly.

INSERT INTO BANDS (band\_id, name, origin\_country, formed\_year)

VALUES (3, 'Louisiana Five', 'United States', 1917);

Msg 547, Level 16, State 0, Line 1

The INSERT statement conflicted with the CHECK constraint "CK\_\_BANDS\_\_formed\_ye\_\_3D5E1FD2". The conflict occurred in database "test\_f9", table "dbo.BANDS", column 'formed\_year'.

The statement has been terminated.

Task 2

1. Creation and selecting database as an active one:

CREATE DATABASE LIBRARY;

USE LIBRARY;

1. Creation of the MEMBERS table:

CREATE TABLE MEMBERS (

CardNo CHAR(5) PRIMARY KEY,

Surname VARCHAR(15) NOT NULL,

Name VARCHAR(15) NOT NULL,

Address VARCHAR(150),

Birthday\_DATE DATE NOT NULL,

Gender CHAR,

Phone\_No VARCHAR(15),

CONSTRAINT CK\_Gender CHECK ([Gender] IN ('M', 'F')),

CONSTRAINT CardNo\_length CHECK ( LEN([CardNo]) = 5 )

);

1. Creation of the Employees table and adding the Gender field:

CREATE TABLE Employees (

emp\_id INTEGER PRIMARY KEY IDENTITY(1,1),

Surname VARCHAR(15) NOT NULL,

Name VARCHAR(15) NOT NULL,

Birthday\_DATE DATE NOT NULL,

Emp\_DATE DATE,

Gender CHAR,

CONSTRAINT CK\_Emp\_DATE CHECK (Emp\_DATE > Birthday\_DATE),

CONSTRAINT CK\_Gender\_Employees CHECK ([Gender] IN ('M', 'F'))

);

1. Creation of the Publishers table:

CREATE TABLE Publishers (

pub\_id INTEGER PRIMARY KEY IDENTITY(1,1),

Name VARCHAR(50) NOT NULL,

City VARCHAR(50) NOT NULL,

Phone\_No VARCHAR(15)

);

1. Creation of the Books table:

CREATE TABLE Books (

BookID CHAR(5) PRIMARY KEY,

Pub\_ID INTEGER FOREIGN KEY REFERENCES Publishers(pub\_id),

Type VARCHAR,

Price MONEY NOT NULL,

Title VARCHAR(40) NOT NULL,

CONSTRAINT BookID\_length CHECK ( LEN([BookID]) = 5 ),

CONSTRAINT CK\_Type CHECK (Type IN ('novel', 'historical', 'for kids', 'poems', 'crime story', 'science fiction', 'science'))

);

1. Creation of the BOOK\_LOANS table and adding constraint forcing the uniqueness of the pair values:

CREATE TABLE BOOK\_LOANS (

LoanID INTEGER PRIMARY KEY IDENTITY(1,1),

CardNo CHAR(5) FOREIGN KEY REFERENCES MEMBERS(CardNo),

BookID CHAR(5) FOREIGN KEY REFERENCES Books(BookID),

emp\_id INTEGER FOREIGN KEY REFERENCES Employees(emp\_id),

DateOut DATE,

DueDate DATE,

Penalty MONEY CHECK (Penalty >= 0) DEFAULT 0,

CONSTRAINT CK\_DATE CHECK (DueDate > DateOut),

);

**Task 3**

1. Determine the structure of all database's tables.



1. Display names and salaries of employees.

SELECT first\_name, last\_name, salary

FROM employees;

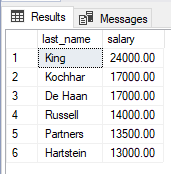


1. Display the last name and salary of employees earning more than $12,000.

SELECT last\_name, salary

FROM employees

WHERE salary>12000;

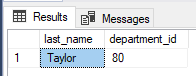


1. Display the last name and department number for employee number 176.

SELECT last\_name, department\_id

FROM employees

WHERE employee\_id = 176;



1. Display the last name and salary for all employees whose salary is not in the range of $5,000 to $12,000.

SELECT last\_name, salary

FROM employees

WHERE NOT(salary<=12000 AND salary>=5000);



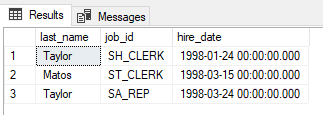
1. Display the last name, job ID, and start date (hire date) for the employees with the last names of Matos and Taylor. Order the query in ascending order by start date.

SELECT last\_name, job\_id, hire\_date

FROM employees

WHERE last\_name='Matos' OR last\_name='Taylor'

ORDER BY hire\_date;



1. Display the last name and department number of all employees in departments 20 or 50 in ascending alphabetical order by name.

SELECT last\_name, department\_id

FROM employees

WHERE department\_id=20 OR department\_id=50

ORDER BY last\_name;



1. Display the last name and job title of all employees who do not have a manager.

SELECT last\_name, job\_title

FROM employees JOIN jobs

ON employees.job\_id = jobs.job\_id

WHERE manager\_id IS NULL;



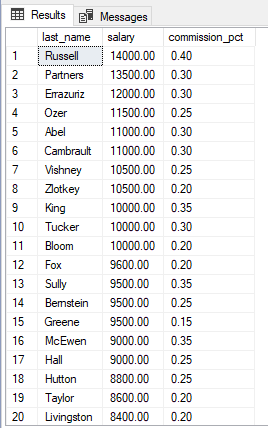
1. Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

SELECT last\_name, salary, commission\_pct

FROM employees

WHERE commission\_pct IS NOT NULL

ORDER BY salary DESC, commission\_pct DESC;



1. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively.

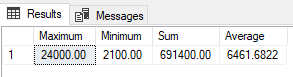
SELECT MAX(salary) AS Maximum,

MIN(salary) AS Minimum,

SUM(salary) AS Sum,

AVG(salary) AS Average

FROM employees;



1. Modify the previous query to display the minimum, maximum, sum, and average salary for each job type (job\_id).

SELECT job\_id,

MAX(salary) AS Maximum,

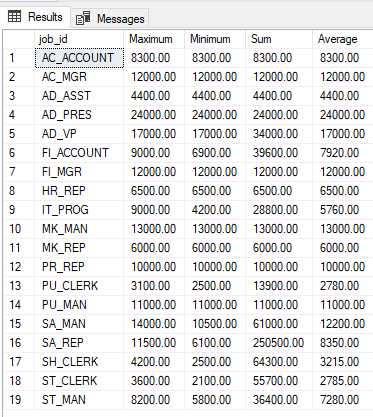
MIN(salary) AS Minimum,

SUM(salary) AS Sum,

AVG(salary) AS Average

FROM employees

GROUP BY job\_id;



1. Display the number of people with the same job.

SELECT SUM(myCol)

FROM (

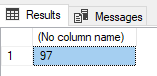
SELECT count(job\_id) AS myCol

FROM employees

GROUP BY job\_id

) AS job\_id\_subquery

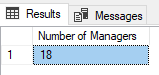
WHERE myCol > 1;



1. Determine the number of managers without listing them. Label the column Number of Managers. Hint: Use the MANAGER\_ID column to determine the number of managers.

SELECT COUNT(DISTINCT manager\_id) AS 'Number of Managers'

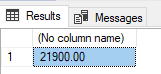
FROM employees;



1. Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

SELECT MAX(salary) - MIN(salary)

FROM employees;



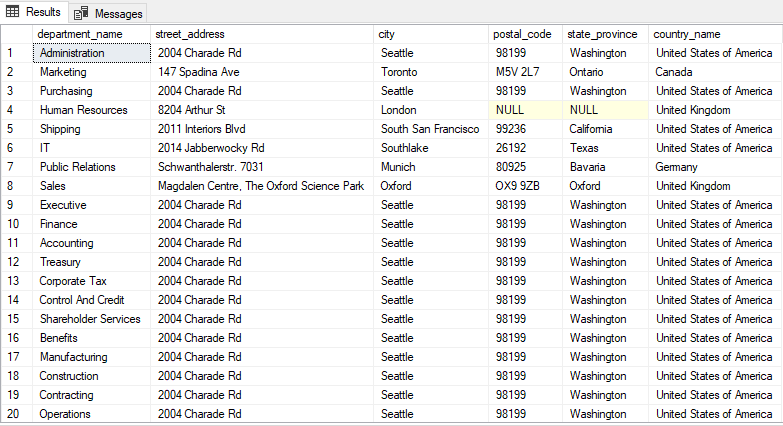
1. Find the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location ID, street address, city, state or province, and country in the output.

Assuming that we want to display the departments’ names as well:

SELECT department\_name, street\_address, city, postal\_code, state\_province, country\_name

FROM departments JOIN (locations JOIN countries ON locations.country\_id = countries.country\_id)

ON departments.location\_id = locations.location\_id;



1. Display the last name and department name for all employees.

SELECT last\_name, department\_name

FROM employees JOIN departments

ON employees.department\_id = departments.department\_id;



1. Display the last name, job, department number, and department name for all employees who work in Toronto.

SELECT last\_name, job\_title, e.department\_id, department\_name

FROM employees AS e

JOIN (departments AS d

JOIN locations AS l

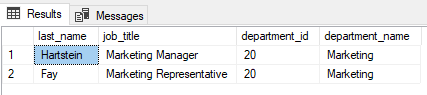
ON d.location\_id = l.location\_id)

ON e.department\_id = d.department\_id

JOIN jobs AS j

ON e.job\_id = j.job\_id

WHERE city='Toronto';



# **Additional exercises**

1. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude and groups where the minimum salary is $6000 or less. Sort the output in descending order of salary.

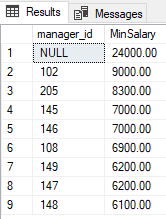
SELECT manager\_id, MIN(salary) AS MinSalary

FROM employees

GROUP BY manager\_id

HAVING MIN(salary) > 6000

ORDER BY MinSalary DESC;



1. The HR department wants to determine the names of all employees who were hired after Davies. Create a query to display the name and hire date of any employee hired after employee Davies.

SELECT first\_name, last\_name, hire\_date

FROM employees

WHERE hire\_date > (

SELECT hire\_date

FROM employees

WHERE last\_name = 'Davies'

);



1. The HR department needs to find the names and hire dates for all employees who were hired before their managers, along with their managers' names and hire dates.

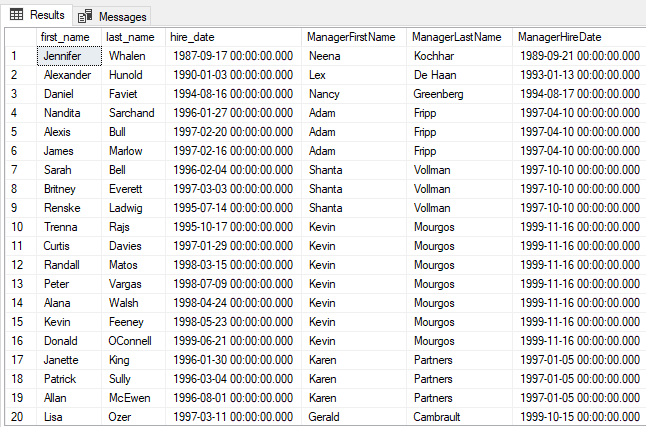
SELECT e1.first\_name, e1.last\_name, e1.hire\_date, e2.first\_name as ManagerFirstName, e2.last\_name as ManagerLastName, e2.hire\_date as ManagerHireDate

FROM employees e1

JOIN employees e2

ON e1.manager\_id = e2.employee\_id

WHERE e1.hire\_date < e2.hire\_date;



1. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

SELECT employee\_id, last\_name, salary

FROM employees

WHERE salary > (

SELECT AVG(salary)

FROM employees

)

ORDER BY salary;



1. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name starts with "U".

SELECT employee\_id, last\_name

FROM employees

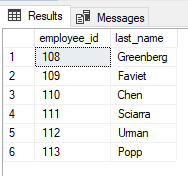
WHERE department\_id = (

SELECT department\_id

FROM employees

WHERE last\_name LIKE 'U%'

);



1. Create a report for HR that displays the last name and salary of every employee who reports to King.

SELECT e1.last\_name, e1.salary

FROM employees e1 JOIN employees e2

ON e1.manager\_id = e2.employee\_id

WHERE e2.last\_name = 'King';



1. For budgeting purposes, the HR department needs a report on projected 10% raises. The report should display those employees who have no commissions.

SELECT last\_name, salary

FROM employees

WHERE commission\_pct IS NULL;

