Task 1

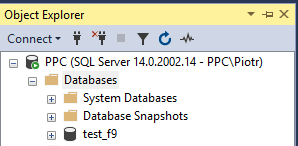
1. Choose New Query option for opening SQL worksheet window.



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),

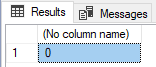
origin\_country VARCHAR(50),

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),

);

# **Additional exercises**

1. Creation and selecting database as active one:

CREATE DATABASE video\_renting;   
USE video\_renting;

1. Creation of Member table:

CREATE TABLE Member (

MEMBER\_ID INTEGER IDENTITY(1, 1) PRIMARY KEY,

LAST\_NAME VARCHAR(25) NOT NULL,

FIRST\_NAME VARCHAR(25),

ADDRESS VARCHAR(100),

CITY VARCHAR(30),

PHONE VARCHAR(15),

JOIN\_DATE DATETIME DEFAULT GETDATE() NOT NULL

);

1. Creation of Title table with Category and Rating as enumerable char values:

CREATE TABLE Title (

TITLE\_ID INTEGER IDENTITY(1, 1) PRIMARY KEY,

TITLE VARCHAR(60) NOT NULL,

DESCRIPTION VARCHAR(400) NOT NULL,

RATING VARCHAR(4) CHECK (RATING IN ('G', 'PG', 'R', 'NC17', 'NR')),

CATEGORY VARCHAR(20) CHECK (CATEGORY IN ('DRAMA', 'COMEDY', 'ACTION', 'CHILD', 'SCIFI', 'DOCUMENTARY')),

RELEASE\_DATE DATETIME

);

1. Creation of Title\_copy table with primary key as composition of own ID with foreign key to Title ID:

CREATE TABLE Title\_copy (

COPY\_ID INTEGER NOT NULL,

TITLE\_ID INTEGER FOREIGN KEY REFERENCES Title(TITLE\_ID) NOT NULL,

RATING VARCHAR(15) CHECK (RATING IN ('AVAILABLE', 'DESTROYED', 'RENTED', 'RESERVED')) NOT NULL,

CONSTRAINT pk\_title\_copy PRIMARY KEY (

COPY\_ID,

TITLE\_ID

)

);

1. Creation of Rental table with a foreign key to Title\_copy primary key which consists of two values:

CREATE TABLE Rental (

BOOK\_DATE DATE DEFAULT GETDATE(),

COPY\_ID INTEGER,

MEMBER\_ID INTEGER FOREIGN KEY REFERENCES Member(MEMBER\_ID),

TITLE\_ID INTEGER,

ACT\_RET\_DATE DATETIME,

EXP\_RET\_DATE DATETIME DEFAULT DATEADD(day, 2, GETDATE()),

CONSTRAINT pk\_rental PRIMARY KEY (

BOOK\_DATE,

MEMBER\_ID,

COPY\_ID

),

CONSTRAINT fk\_rental FOREIGN KEY (

COPY\_ID,

TITLE\_ID

) REFERENCES Title\_copy(COPY\_ID, TITLE\_ID)

);

1. Creation of Reservation table with unique composition of two values:

CREATE TABLE Reservation (

RES\_DATE DATETIME NOT NULL,

MEMBER\_ID INTEGER FOREIGN KEY REFERENCES Member(MEMBER\_ID) NOT NULL,

TITLE\_ID INTEGER FOREIGN KEY REFERENCES Title(TITLE\_ID) NOT NULL,

CONSTRAINT fk\_reservation PRIMARY KEY (

RES\_DATE,

MEMBER\_ID,

TITLE\_ID

),

CONSTRAINT unique\_composition UNIQUE (

RES\_DATE,

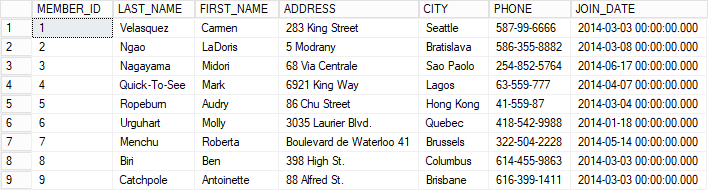
MEMBER\_ID

)

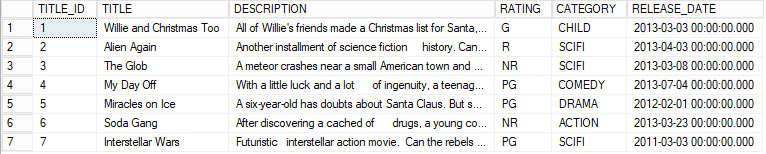
);

1. Results of execution of popul\_video.sql query can be obtained by:

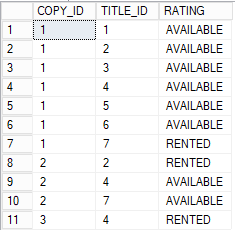
SELECT \* FROM Member;



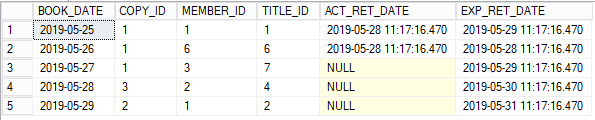
SELECT \* FROM Title;



SELECT \* FROM Title\_copy;



SELECT \* FROM Rental;



SELECT \* FROM Reservation;



**Task 3**

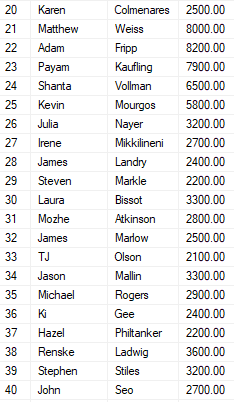
1. Determine the structure of all database's tables.

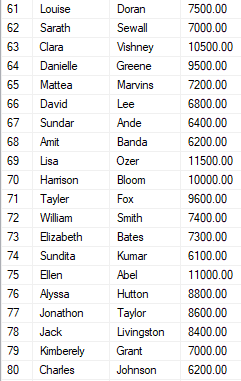


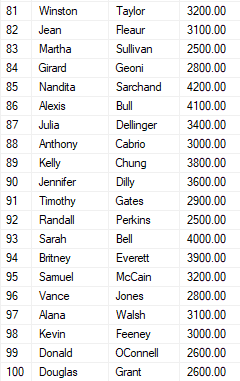
1. Display names and salaries of employees.

SELECT first\_name, last\_name, salary

FROM employees;

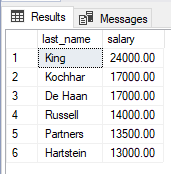
 

1. Display the last name and salary of employees earning more than $12,000.

SELECT last\_name, salary

FROM employees

WHERE salary>12000;

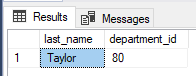


1. Display the last name and department number for employee number 176.

SELECT last\_name, department\_id

FROM employees

WHERE employee\_id = 176;

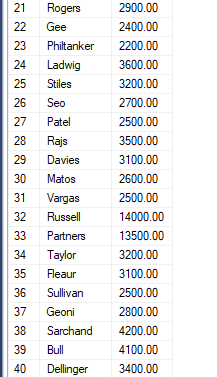
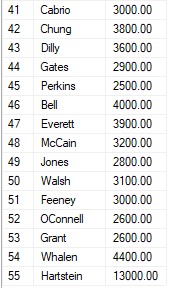


1. Display the last name and salary for all employees whose salary is not in the range of $5,000 to $12,000.

SELECT last\_name, salary

FROM employees

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

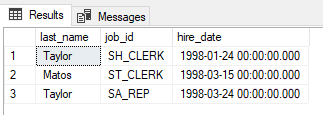
1. Display the last name, job ID, and start date (hire date) for the employees with the last names of Matos and Taylor. Order the query in ascending order by start date.

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

FROM employees

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

ORDER BY hire\_date;



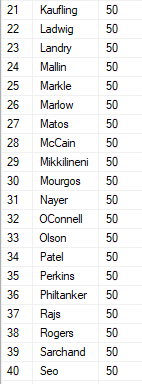
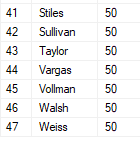
1. Display the last name and department number of all employees in departments 20 or 50 in ascending alphabetical order by name.

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;



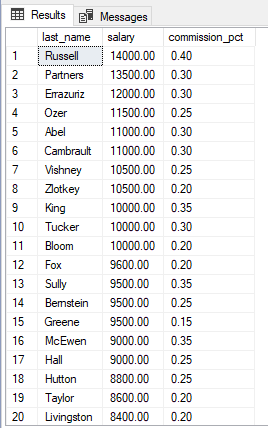
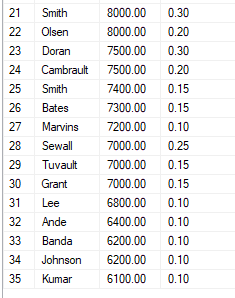
1. Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

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.

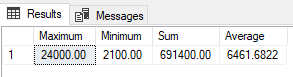
SELECT MAX(salary) AS Maximum,

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,

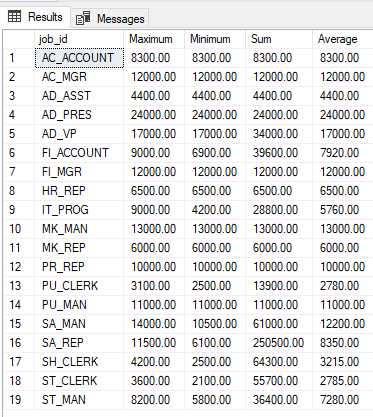
MIN(salary) AS Minimum,

SUM(salary) AS Sum,

AVG(salary) AS Average

FROM employees

GROUP BY job\_id;

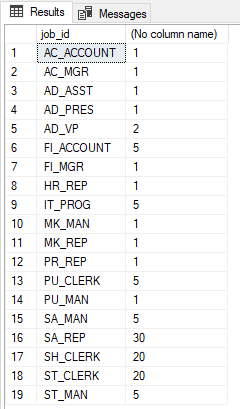


1. Display the number of people with the same job.

SELECT job\_id, count(job\_id)

FROM employees

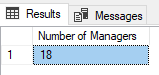
GROUP BY job\_id;



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'

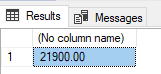
FROM employees;



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

SELECT MAX(salary) - MIN(salary)

FROM employees;

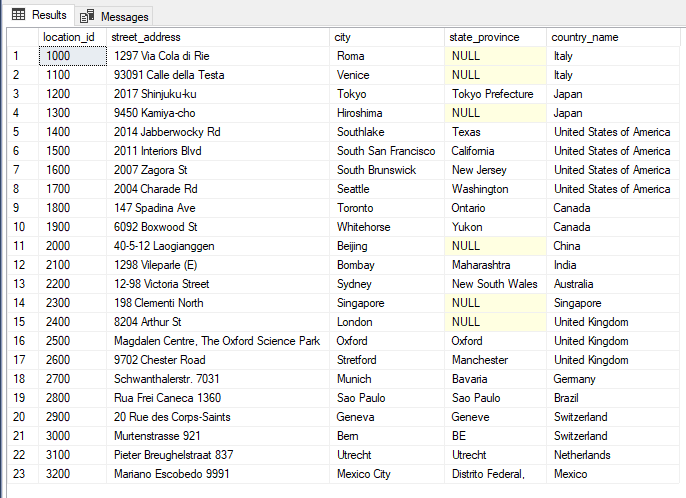


1. Find the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location ID, street address, city, state or province, and country in the output.

SELECT location\_id, street\_address, city, state\_province, country\_name

FROM locations JOIN countries

ON locations.country\_id = countries.country\_id;

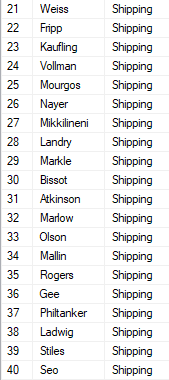
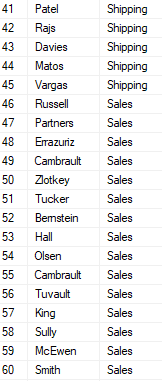
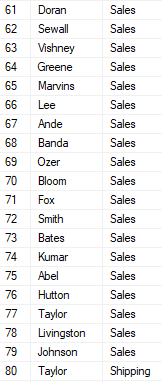


1. Display the last name and department name for all employees.

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, e.job\_id, e.department\_id, department\_name

FROM employees AS e

JOIN (departments AS d

JOIN locations AS l

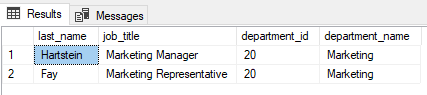
ON d.location\_id = l.location\_id)

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.

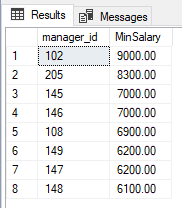
SELECT manager\_id, MIN(salary) AS MinSalary

FROM employees

GROUP BY manager\_id

HAVING MIN(salary) > 6000 AND manager\_id IS NOT NULL

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

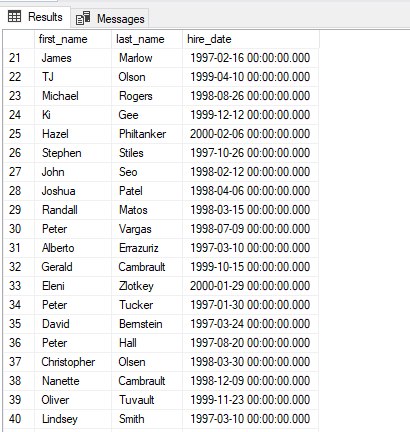
WHERE hire\_date > (

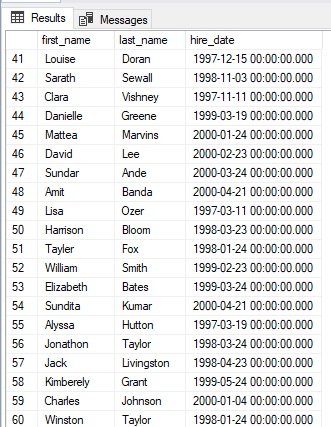
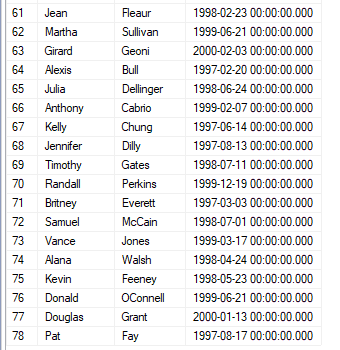
SELECT hire\_date

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.

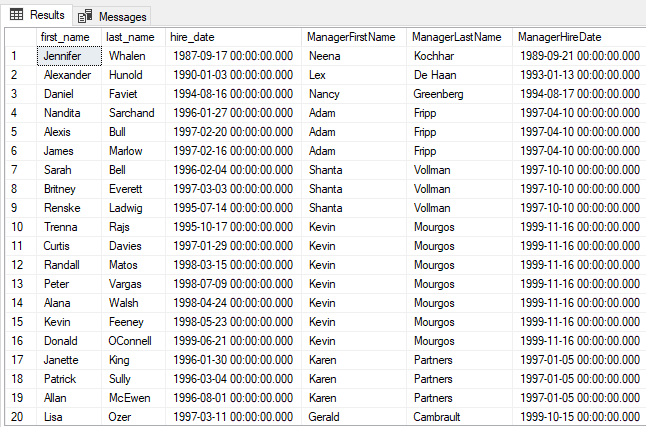
SELECT e1.first\_name, e1.last\_name, e1.hire\_date, e2.first\_name as ManagerFirstName, e2.last\_name as ManagerLastName, e2.hire\_date as ManagerHireDate

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

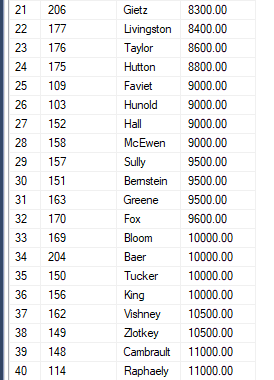
WHERE salary > (

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

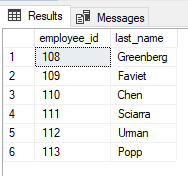
WHERE department\_id = (

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';

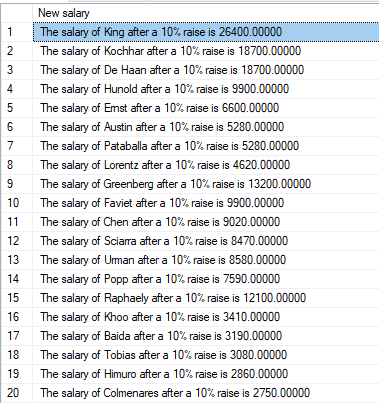
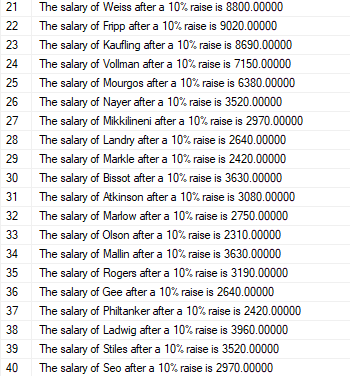


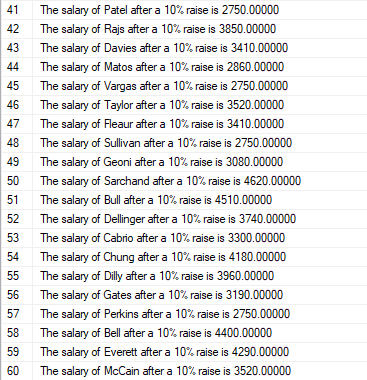
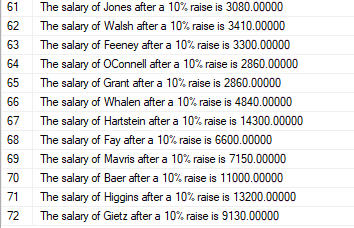
1. For budgeting purposes, the HR department needs a report on projected 10% raises. The report should display those employees who have no commissions.

SELECT 'The salary of ' + last\_name + ' after a 10% raise is ' + CAST(salary\*1.1 AS CHAR) AS 'New salary'

FROM employees

WHERE commission\_pct IS NULL;

**Task** **4**

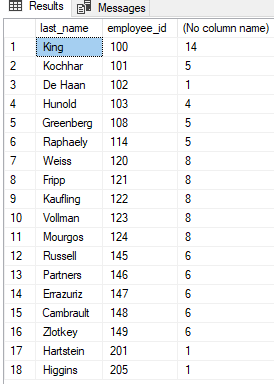
1. Show last names and numbers of all managers together with the number of employees that are his / her subordinates.

SELECT e2.last\_name, e2.employee\_id, COUNT(\*)

FROM employees e1 join employees e2

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

GROUP BY e2.employee\_id, e2.last\_name



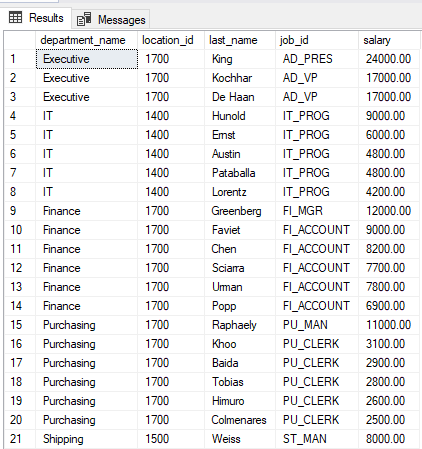
1. Create a report that displays the department name, location name, job title and salary of those employees who work in a specific (given) location.

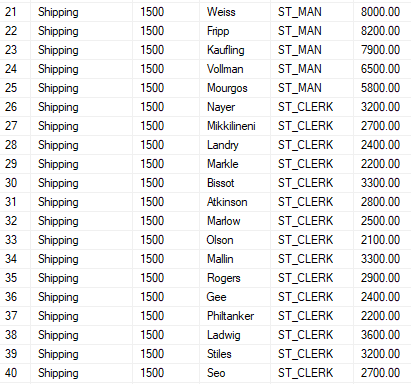
SELECT d1.department\_name, l1.location\_id, e1.last\_name, e1.job\_id, e1.salary

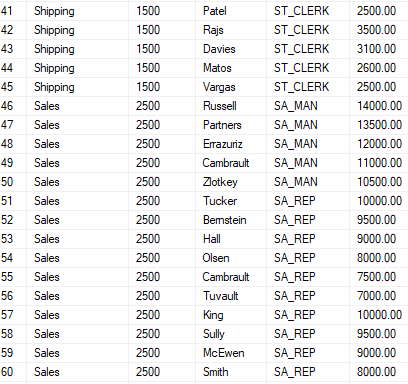
FROM employees e1 join (departments d1 join locations l1

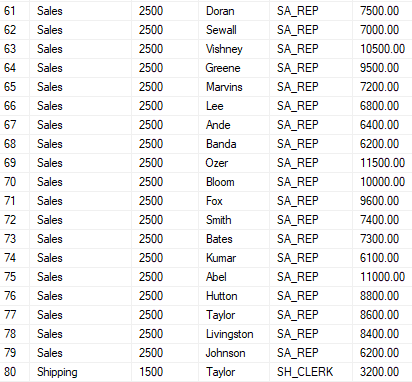
ON d1.location\_id = l1.location\_id)

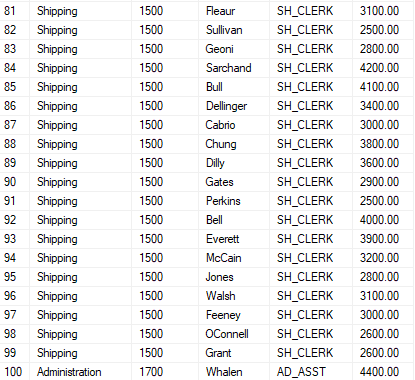
ON e1.department\_id = d1.department\_id

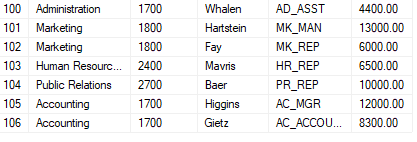










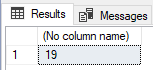


1. Find the number of employees who have a last name that ends with the letter n.

SELECT count(\*)

FROM employees

WHERE last\_name LIKE '%n'



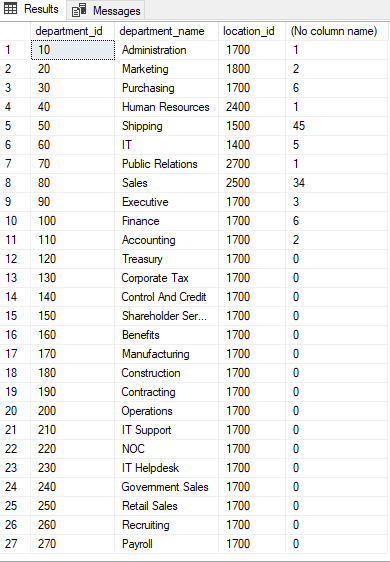
1. Create a report that shows the name, location and the number of employees for each department. Make sure that report also includes departments without employees.

SELECT d1.department\_id, d1.department\_name, d1.location\_id, COUNT(e1.last\_name)

FROM departments d1 left join employees e1

ON d1.department\_id = e1.department\_id

GROUP BY d1.department\_id, d1.department\_name, d1.location\_id

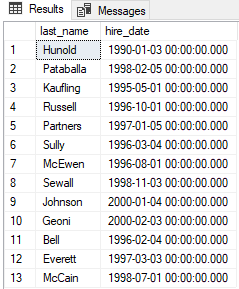


1. Show all employees who were hired in the first five days of the month (before the 6th of the month).

SELECT last\_name, hire\_date

FROM employees

WHERE DAY(hire\_date) < 6



1. Create a report to display the department number and lowest salary of the department with the highest average salary.

SELECT maxAvgSal.department\_id, MIN(e1.salary)

FROM (

SELECT TOP 1 d1.department\_id, AVG(e1.salary) AS avgSal

FROM departments d1

JOIN employees e1 ON d1.department\_id = e1.department\_id

GROUP BY d1.department\_id

ORDER BY avgSal DESC

) AS maxAvgSal

JOIN employees e1 ON maxAvgSal.department\_id = e1.department\_id

GROUP BY maxAvgSal.department\_id



1. Create a report that displays department where no sales representatives work. Include the department number, department name and location in the output.

SELECT d.department\_id, d.department\_name, d.manager\_id, d.location\_id

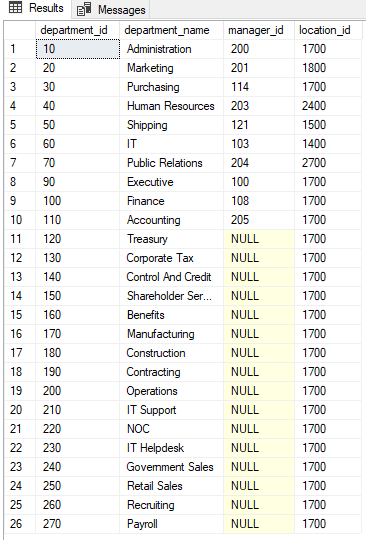
FROM departments d

WHERE d.department\_id NOT IN (

SELECT department\_id

FROM employees

WHERE job\_id='SA\_REP' AND department\_id IS NOT NULL);



1. Display the department number, department name and the number of employees for the department:
2. with the highest number of employees.

WITH emplTable (department\_id, department\_name, numOfEmpl)

AS (

SELECT d1.department\_id, d1.department\_name, count(e1.last\_name) AS numOfEmpl

FROM departments d1

JOIN employees e1 ON d1.department\_id = e1.department\_id

GROUP BY d1.department\_id, d1.department\_name

)

SELECT department\_id, department\_name, numOfEmpl

FROM emplTable

WHERE numOfEmpl = (

SELECT MAX(numOfEmpl)

FROM emplTable

);



1. with the lowest number of employees

WITH emplTable (department\_id, department\_name, numOfEmpl)

AS (

SELECT d1.department\_id, d1.department\_name, count(e1.last\_name) AS numOfEmpl

FROM departments d1

JOIN employees e1 ON d1.department\_id = e1.department\_id

GROUP BY d1.department\_id, d1.department\_name

)

SELECT department\_id, department\_name, numOfEmpl

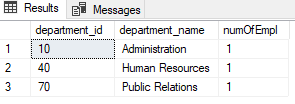
FROM emplTable

WHERE numOfEmpl = (

SELECT MIN(numOfEmpl)

FROM emplTable

);



1. that employs fewer than three employees.

WITH emplTable (department\_id, department\_name, numOfEmpl)

AS (

SELECT d1.department\_id, d1.department\_name, count(e1.last\_name) AS numOfEmpl

FROM departments d1

JOIN employees e1 ON d1.department\_id = e1.department\_id

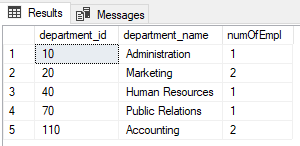
GROUP BY d1.department\_id, d1.department\_name

)

SELECT department\_id, department\_name, numOfEmpl

FROM emplTable

WHERE numOfEmpl < 3;



1. Display years and total numbers of employees that were employed in that year.

DECLARE @StartYear AS INT

DECLARE @EndYear AS INT

DECLARE @CurrentYear AS INT

--Find the year in which the first employee(s) was (were) hired

SELECT @StartYear = MIN(hireYear)

FROM (

SELECT MIN(YEAR(start\_date)) AS hireYear

FROM job\_history

UNION

SELECT MIN(YEAR(hire\_date)) AS hireYear

FROM employees

) AS minYear

SET @EndYear = YEAR(GETDATE())

SET @CurrentYear = @StartYear

CREATE TABLE #yearsEmployees ([year] INT, [count] INT);

WHILE (@CurrentYear <= @EndYear)

BEGIN

INSERT INTO #yearsEmployees

SELECT @CurrentYear AS Year, COUNT(\*) AS numOfEmpl

FROM (

SELECT hire\_date AS hireDate, GETDATE() AS endDate

FROM employees

UNION

SELECT start\_date AS hireDate, end\_date AS endDate

FROM job\_history

) AS hireDates

WHERE @CurrentYear >= YEAR(hireDate)

AND @CurrentYear <= YEAR(endDate);

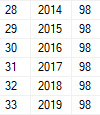
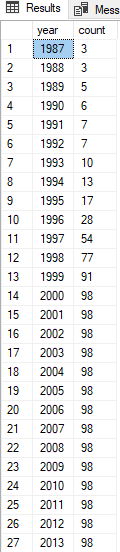
SET @CurrentYear = @CurrentYear + 1;

END

SELECT \*

FROM #yearsEmployees

DROP TABLE #yearsEmployees



1. Display countries and number of locations in that country.

SELECT country\_name, count(l1.country\_id)

FROM countries c1 join locations l1

ON c1.country\_id = l1.country\_id

GROUP BY country\_name



# **Additional** **exercises**

1. Create a query to display the employees who earn a salary that is higher than the salary of all the sales managers (JOB\_ID = 'SA\_MAN'). Sort the results from the highest to the lowest.

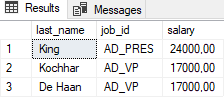
SELECT last\_name, job\_id, salary

FROM employees

WHERE salary > (SELECT MAX(salary)

FROM employees

WHERE job\_id = 'SA\_MAN');



1. Display details such as the employee ID, last name, and department ID of those employees who works in cities the names of which begin with 'T'.

SELECT employee\_id, last\_name, e1.department\_id

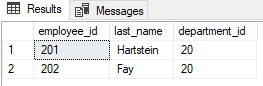
FROM employees e1

JOIN (

departments d1 JOIN locations l1 ON d1.location\_id = l1.location\_id

) ON e1.department\_id = d1.department\_id

WHERE l1.city LIKE 'T%'



1. Write a query to find all employees who earn more than the average salary in their department.

SELECT e1.last\_name, e1.salary, e1.department\_id, avgSalDept.avgDept

FROM employees e1

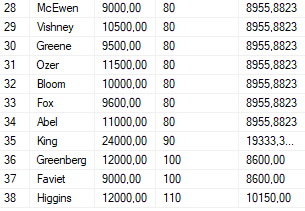
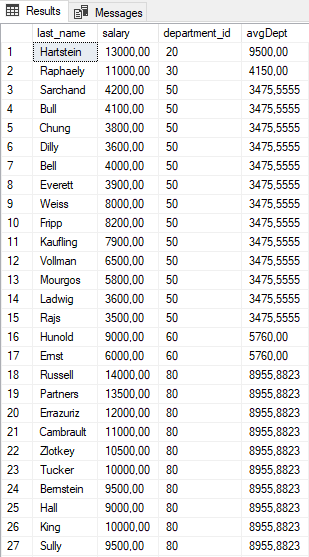
JOIN (SELECT department\_id, AVG(salary) AS avgDept

FROM employees

GROUP BY department\_id

) AS avgSalDept ON e1.department\_id = avgSalDept.department\_id

WHERE e1.salary > avgDept



1. Find all employees who are not supervisors (managers). Do this using the NOT EXISTS operator.

SELECT EMPL.last\_name

FROM employees EMPL

WHERE NOT EXISTS (

SELECT MGRS\_TUPLE\_LIST.employee\_id

FROM employees MGRS\_TUPLE\_LIST

JOIN (

SELECT e1.manager\_id

FROM employees e1

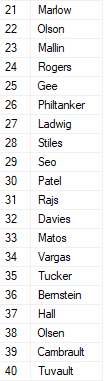
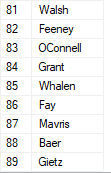
GROUP BY e1.manager\_id

HAVING e1.manager\_id IS NOT NULL

) AS MGRS\_ID\_LIST ON MGRS\_TUPLE\_LIST.employee\_id = MGRS\_ID\_LIST.manager\_id

AND EMPL.employee\_id = MGRS\_TUPLE\_LIST.employee\_id

);

Can it be done using NOT IN?

SELECT EMPL.last\_name

FROM employees EMPL

WHERE EMPL.employee\_id NOT IN (

SELECT MGRS\_TUPLE\_LIST.employee\_id

FROM employees MGRS\_TUPLE\_LIST

JOIN (

SELECT e1.manager\_id

FROM employees e1

GROUP BY e1.manager\_id

HAVING e1.manager\_id IS NOT NULL

) AS MGRS\_ID\_LIST ON MGRS\_TUPLE\_LIST.employee\_id = MGRS\_ID\_LIST.manager\_id

);

The result is the same as above.

1. Display the last names of the employees who earn less than the average salary in their departments.

SELECT last\_name

FROM employees e1

JOIN (

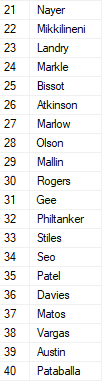
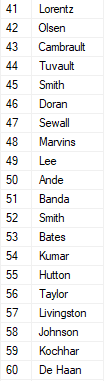
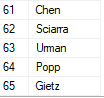
SELECT department\_id, AVG(salary) AS avg\_dept

FROM employees

GROUP BY department\_id

) AS avg\_dept\_list ON e1.department\_id = avg\_dept\_list.department\_id

WHERE salary < avg\_dept

1. Display the last names of the employees who have one or more co-workers in their departments with later hire dates but higher salaries.

SELECT last\_name

FROM (

SELECT DISTINCT e1.last\_name, e1.employee\_id

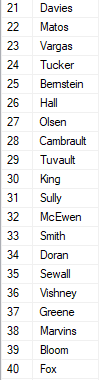
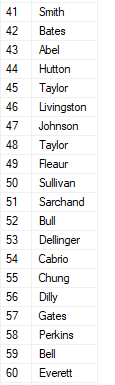
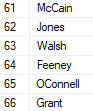
FROM employees e1

JOIN employees e2 ON e1.department\_id = e2.department\_id

WHERE e2.hire\_date > e1.hire\_date

AND e2.salary > e1.salary

) AS emplCol;

1. Display the department names of those departments whose total salary cost is above one-eight (1/8) of the total salary cost of the whole company. Use the WITH clause to write this query. Name the query SUMMARY.

WITH Summary

AS (

SELECT d1.department\_name, SUM(e1.salary) AS dept\_total

FROM departments d1

JOIN employees e1 ON d1.department\_id = e1.department\_id

GROUP BY d1.department\_name

)

SELECT department\_name, dept\_total

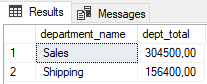
FROM Summary

WHERE dept\_total > (

SELECT SUM(salary) / 8

FROM employees

);



1. Delete the oldest JOB\_HISTORY row of an employee by looking up the JOB\_HISTORY table for the MIN(START\_DATE) for the employee. Delete the records of only those employees who have changed at least two jobs.

WITH hist1 AS (

SELECT employee\_id, MIN(start\_date) AS minStDate, COUNT(\*) AS chgCount

FROM job\_history

GROUP BY employee\_id

),

hist2 AS (

SELECT jh1.employee\_id, MIN(start\_date) AS stDate

FROM job\_history jh1

JOIN hist1 ON jh1.employee\_id = hist1.employee\_id

WHERE chgCount >= 2

GROUP BY jh1.employee\_id

)

SELECT \*

FROM job\_history

WHERE employee\_id IN (

SELECT employee\_id

FROM hist2

)

AND start\_date IN (

SELECT stDate

FROM hist2

);

