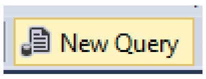
Task 1

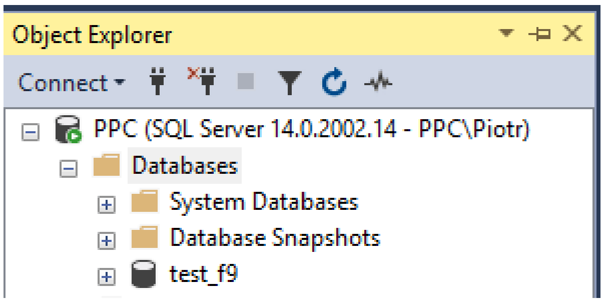
1. Choose New Query option for opening SQL worksheet window.



2. Define new database named test_yourname using CREATE DATABASE statement.

```
CREATE DATABASE test_f9;
```

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



4. Check the name of the database you are connected to. You can change a current database using the statement: USE database_name

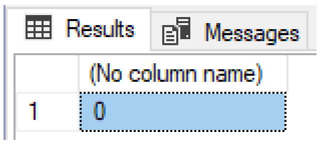
```
USE test_f9;
```

5. 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
       );
```

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

```
SELECT COUNT(*) FROM BANDS;
```



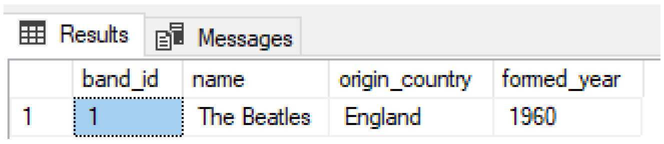
```
Results   Messages
     (No column name)
1    0
```

7. 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);
```

8. Display all the data using SELECT statement.

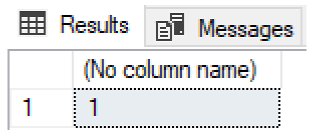
```
SELECT * FROM BANDS;
```



```
Results   Messages
     band_id  name         origin_country  formed_year
1    1        The Beatles  England         1960
```

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

```
SELECT COUNT(*) FROM BANDS;
```



```
Results   Messages
     (No column name)
1    1
```

10. 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),
       );
```

11. 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);
```

12. 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');
```

13. 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);
```

14. 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');
```

15. Add constraint, which doesn't allow entering year earlier than 1920.

```
ALTER TABLE BANDS ADD CHECK (formed_year >= 1920);
```

16. 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;
```

2. 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 )
);
```

3. 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'))
);
```

4. 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)
);
```

5. 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'))
);
```

6. 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;
```

2. 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
       );
```

3. 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
       );
```

4. 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
              )
       );
```

5. Creation of Rental table with a foreign key to Title_copy primary key which consists of two values:

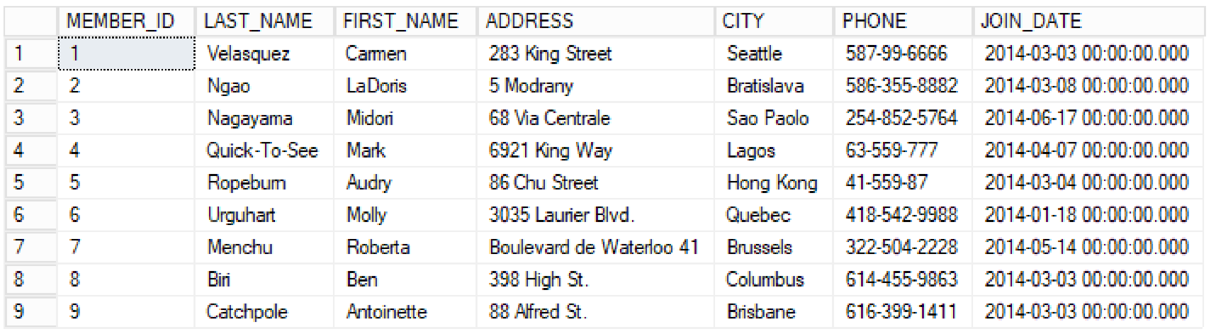
```
CREATE TABLE Rental (
       BOOK_DATE DATE DEFAULT GETDATE(),
       COPY_ID INTEGER,
       MEMBER_ID INTEGER FOREIGN KEY REFERENCES Member(MEMBER_ID),
       TITLE_ID INTEGER,
       ACT_RET_DATE DATETIME,
       EXP_RET_DATE DATETIME DEFAULT DATEADD(day, 2, GETDATE()),
       CONSTRAINT pk_rental PRIMARY KEY (
              BOOK_DATE,
              MEMBER_ID,
              COPY_ID
              ),
       CONSTRAINT fk_rental FOREIGN KEY (
              COPY_ID,
              TITLE_ID
              ) REFERENCES Title_copy(COPY_ID, TITLE_ID)
       );
```

6. 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
              )
       );
```

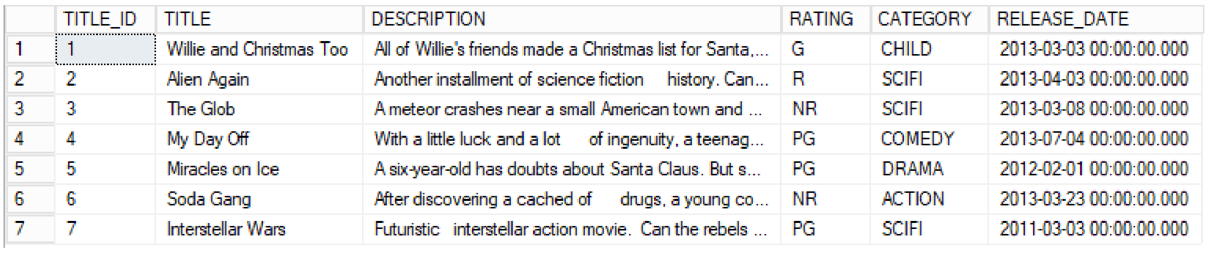
7. Results of execution of popul_video.sql query can be obtained by:

```
SELECT * FROM Member;
```



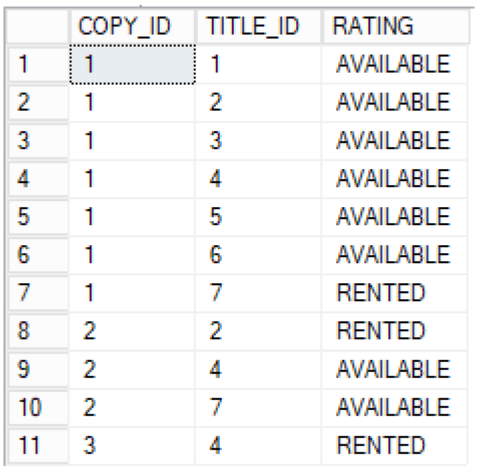
	MEMBER_ID	LAST_NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN_DATE
1	1	Velasquez	Carmen	283 King Street	Seattle	587-99-6666	2014-03-03 00:00:00.000
2	2	Ngao	LaDoris	5 Modrany	Bratislava	586-355-8882	2014-03-08 00:00:00.000
3	3	Nagayama	Midori	68 Via Centrale	Sao Paolo	254-852-5764	2014-06-17 00:00:00.000
4	4	Quick-To-See	Mark	6921 King Way	Lagos	63-559-777	2014-04-07 00:00:00.000
5	5	Ropeburn	Audry	86 Chu Street	Hong Kong	41-559-87	2014-03-04 00:00:00.000
6	6	Urguhart	Molly	3035 Laurier Blvd.	Quebec	418-542-9988	2014-01-18 00:00:00.000
7	7	Menchu	Roberta	Boulevard de Waterloo 41	Brussels	322-504-2228	2014-05-14 00:00:00.000
8	8	Biri	Ben	398 High St.	Columbus	614-455-9863	2014-03-03 00:00:00.000
9	9	Catchpole	Antoinette	88 Alfred St.	Brisbane	616-399-1411	2014-03-03 00:00:00.000

```
SELECT * FROM Title;
```



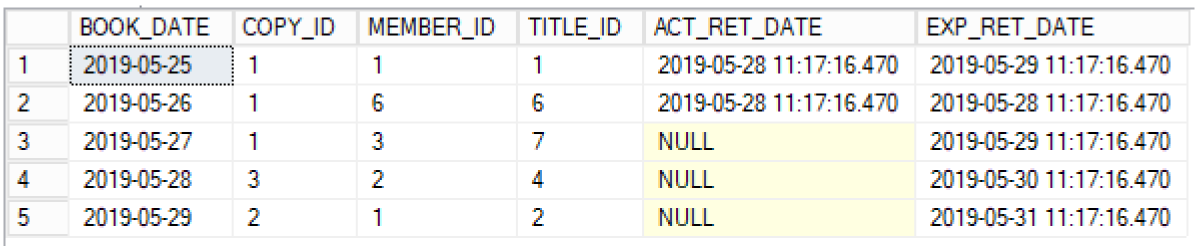
	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_DATE
1	1	Willie and Christmas Too	All of Willie's friends made a Christmas list for Santa,...	G	CHILD	2013-03-03 00:00:00.000
2	2	Alien Again	Another installment of science fiction history. Can...	R	SCIFI	2013-04-03 00:00:00.000
3	3	The Glob	A meteor crashes near a small American town and ...	NR	SCIFI	2013-03-08 00:00:00.000
4	4	My Day Off	With a little luck and a lot of ingenuity, a teenag...	PG	COMEDY	2013-07-04 00:00:00.000
5	5	Miracles on Ice	A six-year-old has doubts about Santa Claus. But s...	PG	DRAMA	2012-02-01 00:00:00.000
6	6	Soda Gang	After discovering a cached of drugs, a young co...	NR	ACTION	2013-03-23 00:00:00.000
7	7	Interstellar Wars	Futuristic interstellar action movie. Can the rebels ...	PG	SCIFI	2011-03-03 00:00:00.000

```
SELECT * FROM Title_copy;
```



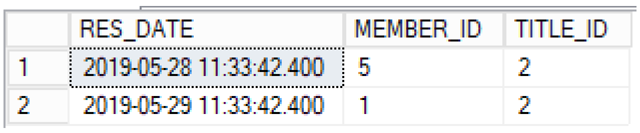
	COPY_ID	TITLE_ID	RATING
1	1	1	AVAILABLE
2	1	2	AVAILABLE
3	1	3	AVAILABLE
4	1	4	AVAILABLE
5	1	5	AVAILABLE
6	1	6	AVAILABLE
7	1	7	RENTED
8	2	2	RENTED
9	2	4	AVAILABLE
10	2	7	AVAILABLE
11	3	4	RENTED

```
SELECT * FROM Rental;
```



	BOOK_DATE	COPY_ID	MEMBER_ID	TITLE_ID	ACT_RET_DATE	EXP_RET_DATE
1	2019-05-25	1	1	1	2019-05-28 11:17:16.470	2019-05-29 11:17:16.470
2	2019-05-26	1	6	6	2019-05-28 11:17:16.470	2019-05-28 11:17:16.470
3	2019-05-27	1	3	7	NULL	2019-05-29 11:17:16.470
4	2019-05-28	3	2	4	NULL	2019-05-30 11:17:16.470
5	2019-05-29	2	1	2	NULL	2019-05-31 11:17:16.470

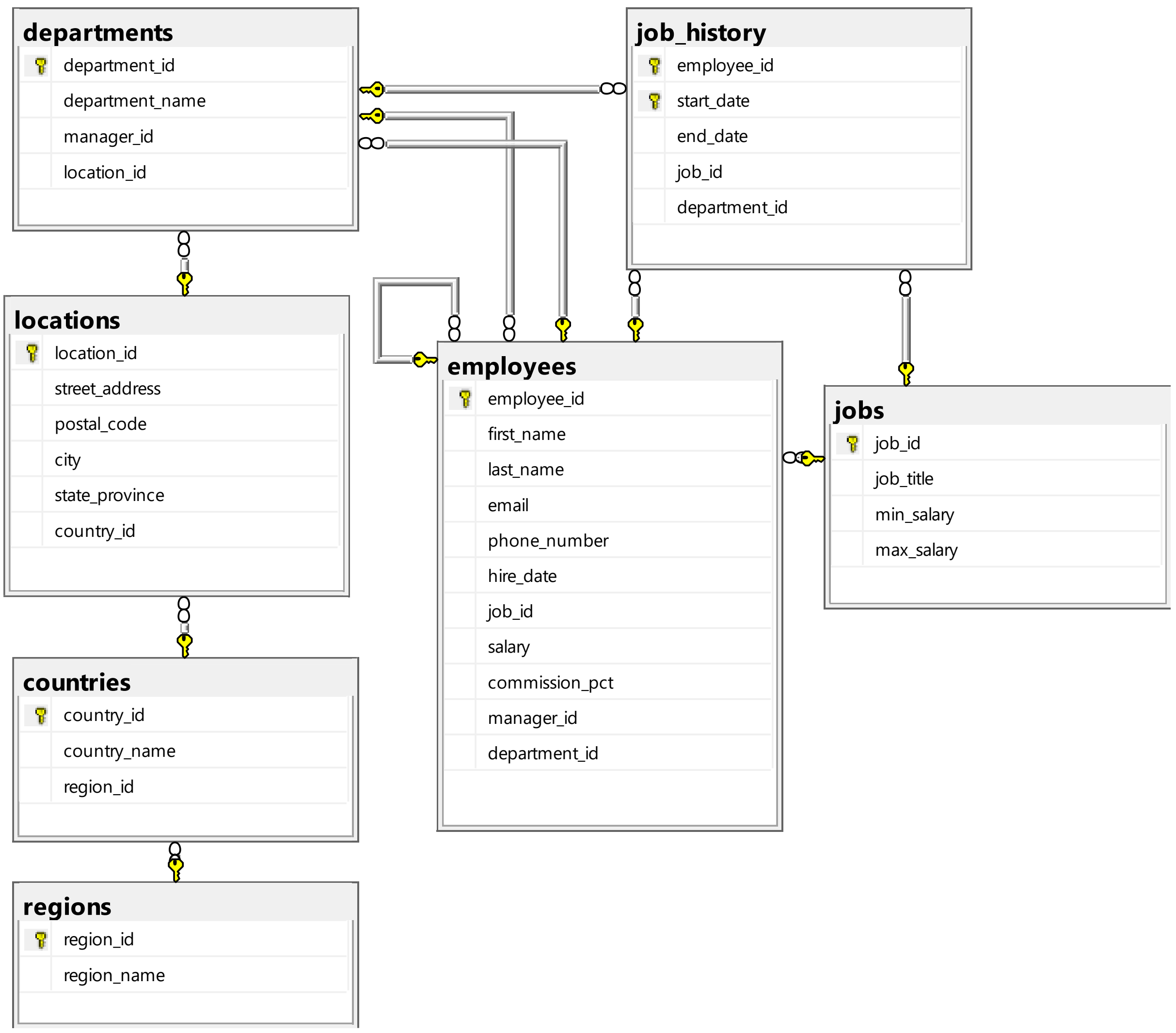
```
SELECT * FROM Reservation;
```



	RES_DATE	MEMBER_ID	TITLE_ID
1	2019-05-28 11:33:42.400	5	2
2	2019-05-29 11:33:42.400	1	2

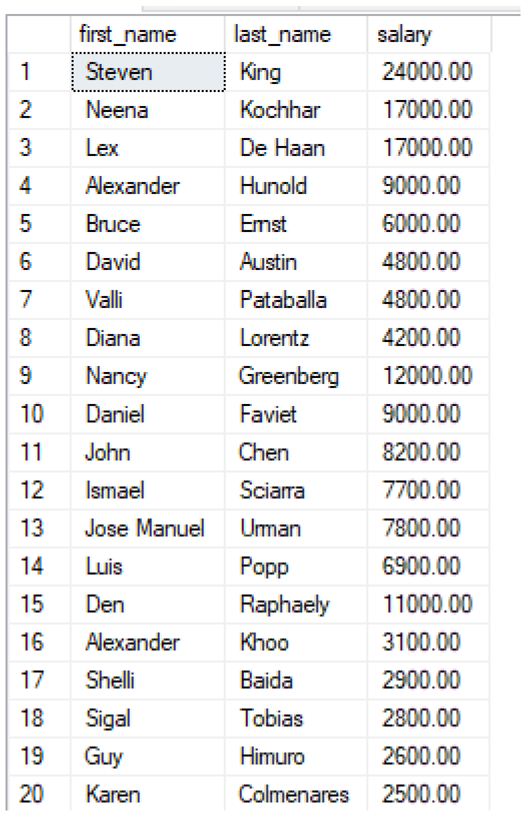
Task 3

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



2. Display names and salaries of employees.

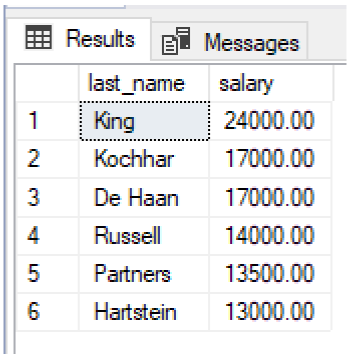
```sql
SELECT first_name, last_name, salary
FROM employees;
```



	first_name	last_name	salary
1	Steven	King	24000.00
2	Neena	Kochhar	17000.00
3	Lex	De Haan	17000.00
4	Alexander	Hunold	9000.00
5	Bruce	Ernst	6000.00
6	David	Austin	4800.00
7	Valli	Pataballa	4800.00
8	Diana	Lorentz	4200.00
9	Nancy	Greenberg	12000.00
10	Daniel	Faviet	9000.00
11	John	Chen	8200.00
12	Ismael	Sciarra	7700.00
13	Jose Manuel	Urman	7800.00
14	Luis	Popp	6900.00
15	Den	Raphaely	11000.00
16	Alexander	Khoo	3100.00
17	Shelli	Baida	2900.00
18	Sigal	Tobias	2800.00
19	Guy	Himuro	2600.00
20	Karen	Colmenares	2500.00

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

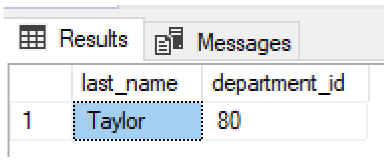
```
SELECT last_name, salary
FROM employees
WHERE salary>12000;
```



	last_name	salary
1	King	24000.00
2	Kochhar	17000.00
3	De Haan	17000.00
4	Russell	14000.00
5	Partners	13500.00
6	Hartstein	13000.00

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

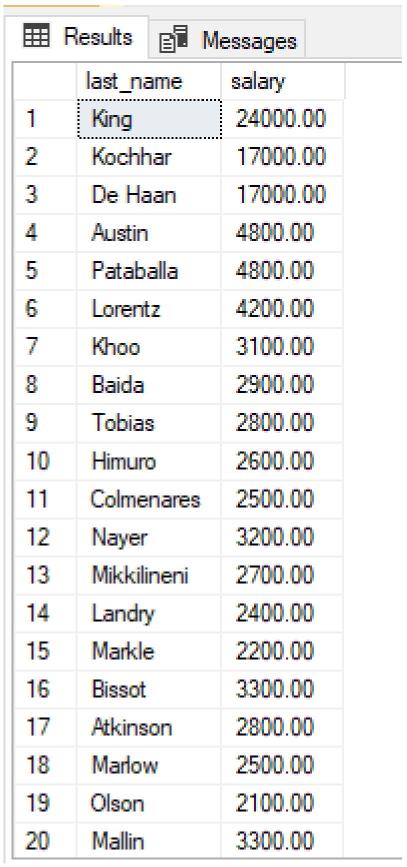
```
SELECT last_name, department_id
FROM employees
WHERE employee_id = 176;
```



	last_name	department_id
1	Taylor	80

5. 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);
```

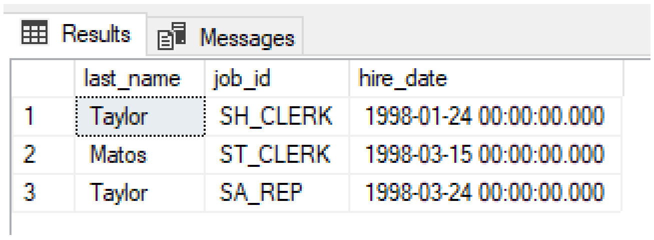


Results | Messages

	last_name	salary
1	King	24000.00
2	Kochhar	17000.00
3	De Haan	17000.00
4	Austin	4800.00
5	Pataballa	4800.00
6	Lorentz	4200.00
7	Khoo	3100.00
8	Baida	2900.00
9	Tobias	2800.00
10	Himuro	2600.00
11	Colmenares	2500.00
12	Nayer	3200.00
13	Mikkilineni	2700.00
14	Landry	2400.00
15	Markle	2200.00
16	Bissot	3300.00
17	Atkinson	2800.00
18	Marlow	2500.00
19	Olson	2100.00
20	Mallin	3300.00

6. 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;
```

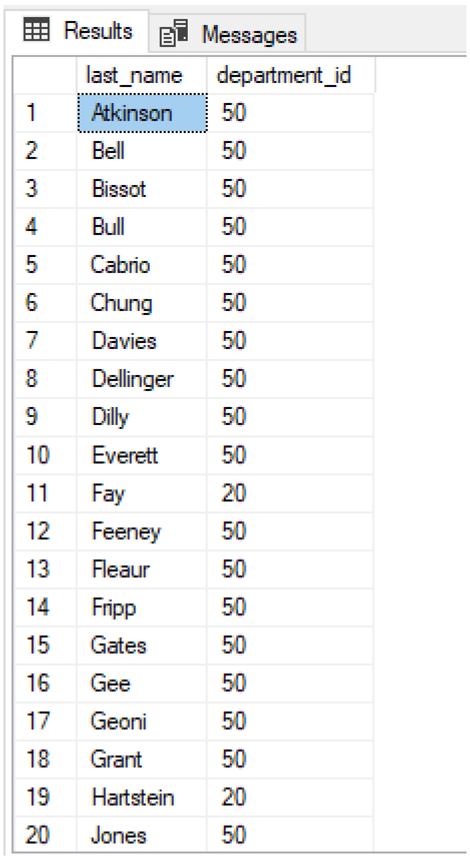


Results | Messages

	last_name	job_id	hire_date
1	Taylor	SH_CLERK	1998-01-24 00:00:00.000
2	Matos	ST_CLERK	1998-03-15 00:00:00.000
3	Taylor	SA_REP	1998-03-24 00:00:00.000

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

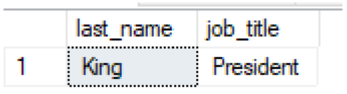
```
SELECT last_name, department_id
FROM employees
WHERE department_id=20 OR department_id=50
ORDER BY last_name;
```



	last_name	department_id
1	Atkinson	50
2	Bell	50
3	Bissot	50
4	Bull	50
5	Cabrio	50
6	Chung	50
7	Davies	50
8	Dellinger	50
9	Dilly	50
10	Everett	50
11	Fay	20
12	Feeney	50
13	Fleaur	50
14	Fripp	50
15	Gates	50
16	Gee	50
17	Geoni	50
18	Grant	50
19	Hartstein	20
20	Jones	50

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

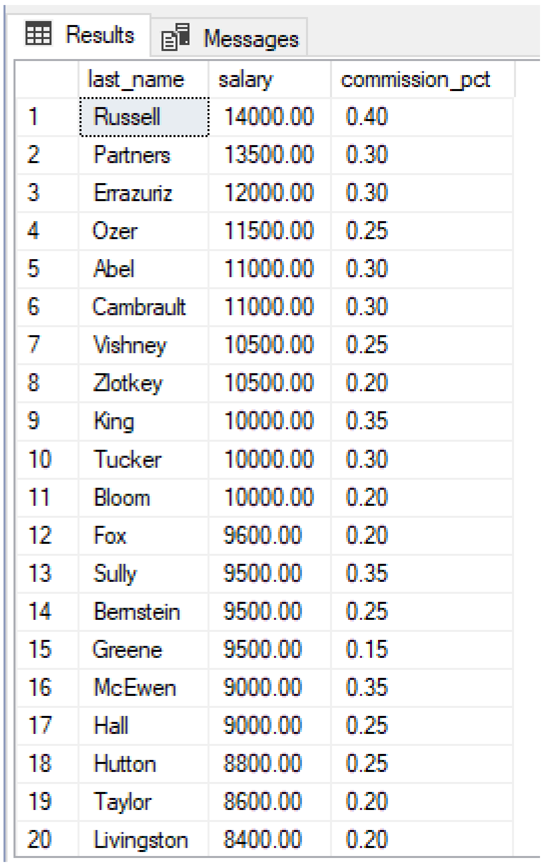
```
SELECT last_name, job_title
FROM employees JOIN jobs
ON employees.job_id = jobs.job_id
WHERE manager_id IS NULL;
```



	last_name	job_title
1	King	President

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

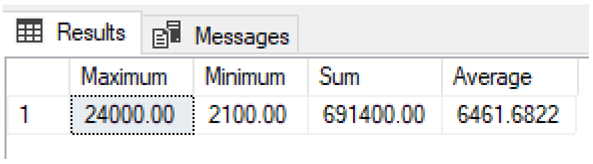
```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL
ORDER BY salary DESC, commission_pct DESC;
```



	last_name	salary	commission_pct
1	Russell	14000.00	0.40
2	Partners	13500.00	0.30
3	Errazuriz	12000.00	0.30
4	Ozer	11500.00	0.25
5	Abel	11000.00	0.30
6	Cambrault	11000.00	0.30
7	Vishney	10500.00	0.25
8	Zlotkey	10500.00	0.20
9	King	10000.00	0.35
10	Tucker	10000.00	0.30
11	Bloom	10000.00	0.20
12	Fox	9600.00	0.20
13	Sully	9500.00	0.35
14	Bernstein	9500.00	0.25
15	Greene	9500.00	0.15
16	McEwen	9000.00	0.35
17	Hall	9000.00	0.25
18	Hutton	8800.00	0.25
19	Taylor	8600.00	0.20
20	Livingston	8400.00	0.20

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

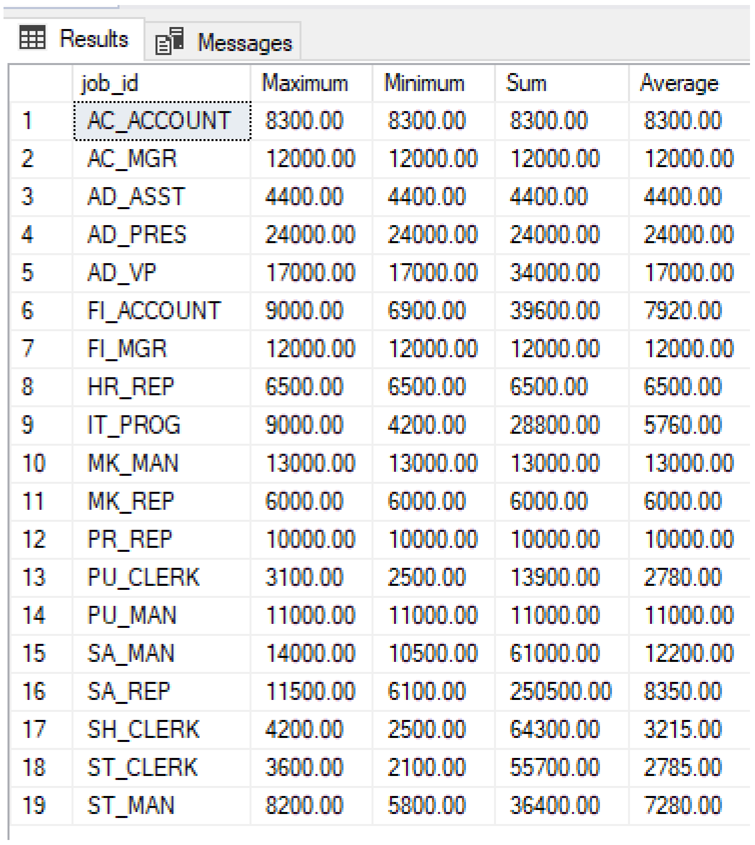
```
SELECT MAX(salary) AS Maximum,
       MIN(salary) AS Minimum,
       SUM(salary) AS Sum,
       AVG(salary) AS Average
FROM employees;
```



	Maximum	Minimum	Sum	Average
1	24000.00	2100.00	691400.00	6461.6822

11. 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;
```

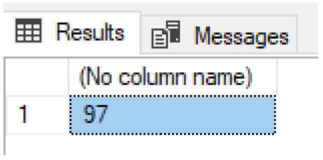


Results | Messages

	job_id	Maximum	Minimum	Sum	Average
1	AC_ACCOUNT	8300.00	8300.00	8300.00	8300.00
2	AC_MGR	12000.00	12000.00	12000.00	12000.00
3	AD_ASST	4400.00	4400.00	4400.00	4400.00
4	AD_PRES	24000.00	24000.00	24000.00	24000.00
5	AD_VP	17000.00	17000.00	34000.00	17000.00
6	FI_ACCOUNT	9000.00	6900.00	39600.00	7920.00
7	FI_MGR	12000.00	12000.00	12000.00	12000.00
8	HR_REP	6500.00	6500.00	6500.00	6500.00
9	IT_PROG	9000.00	4200.00	28800.00	5760.00
10	MK_MAN	13000.00	13000.00	13000.00	13000.00
11	MK_REP	6000.00	6000.00	6000.00	6000.00
12	PR_REP	10000.00	10000.00	10000.00	10000.00
13	PU_CLERK	3100.00	2500.00	13900.00	2780.00
14	PU_MAN	11000.00	11000.00	11000.00	11000.00
15	SA_MAN	14000.00	10500.00	61000.00	12200.00
16	SA_REP	11500.00	6100.00	250500.00	8350.00
17	SH_CLERK	4200.00	2500.00	64300.00	3215.00
18	ST_CLERK	3600.00	2100.00	55700.00	2785.00
19	ST_MAN	8200.00	5800.00	36400.00	7280.00

12. 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;
```

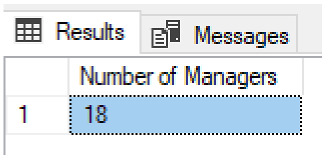


Results | Messages

	(No column name)
1	97

13. 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;
```

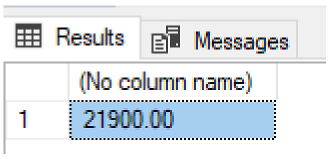


Results | Messages

	Number of Managers
1	18

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

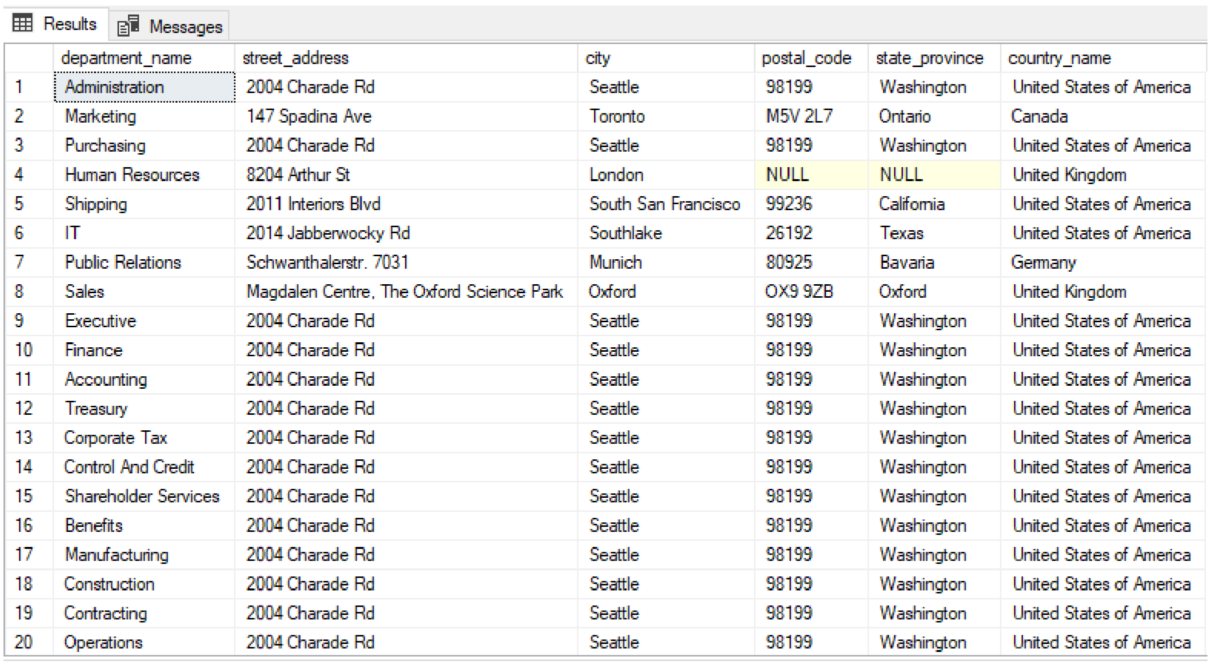
```
SELECT MAX(salary) - MIN(salary)
FROM employees;
```



	(No column name)
1	21900.00

15. 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.

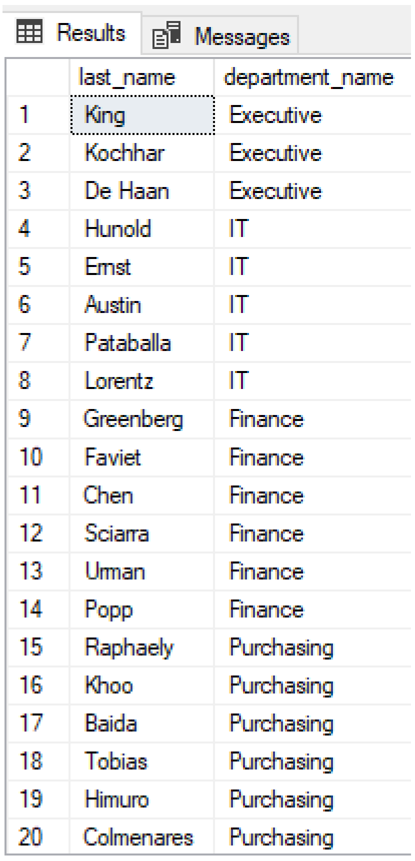
```
Assuming that we want to display the departments' names as well:
SELECT department_name, street_address, city, postal_code, state_province,
country_name
FROM departments JOIN (locations JOIN countries ON locations.country_id =
countries.country_id)
ON departments.location_id = locations.location_id;
```



	department_name	street_address	city	postal_code	state_province	country_name
1	Administration	2004 Charade Rd	Seattle	98199	Washington	United States of America
2	Marketing	147 Spadina Ave	Toronto	M5V 2L7	Ontario	Canada
3	Purchasing	2004 Charade Rd	Seattle	98199	Washington	United States of America
4	Human Resources	8204 Arthur St	London	NULL	NULL	United Kingdom
5	Shipping	2011 Interiors Blvd	South San Francisco	99236	California	United States of America
6	IT	2014 Jabberwocky Rd	Southlake	26192	Texas	United States of America
7	Public Relations	Schwanthalerstr. 7031	Munich	80925	Bavaria	Germany
8	Sales	Magdalen Centre, The Oxford Science Park	Oxford	OX9 9ZB	Oxford	United Kingdom
9	Executive	2004 Charade Rd	Seattle	98199	Washington	United States of America
10	Finance	2004 Charade Rd	Seattle	98199	Washington	United States of America
11	Accounting	2004 Charade Rd	Seattle	98199	Washington	United States of America
12	Treasury	2004 Charade Rd	Seattle	98199	Washington	United States of America
13	Corporate Tax	2004 Charade Rd	Seattle	98199	Washington	United States of America
14	Control And Credit	2004 Charade Rd	Seattle	98199	Washington	United States of America
15	Shareholder Services	2004 Charade Rd	Seattle	98199	Washington	United States of America
16	Benefits	2004 Charade Rd	Seattle	98199	Washington	United States of America
17	Manufacturing	2004 Charade Rd	Seattle	98199	Washington	United States of America
18	Construction	2004 Charade Rd	Seattle	98199	Washington	United States of America
19	Contracting	2004 Charade Rd	Seattle	98199	Washington	United States of America
20	Operations	2004 Charade Rd	Seattle	98199	Washington	United States of America

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

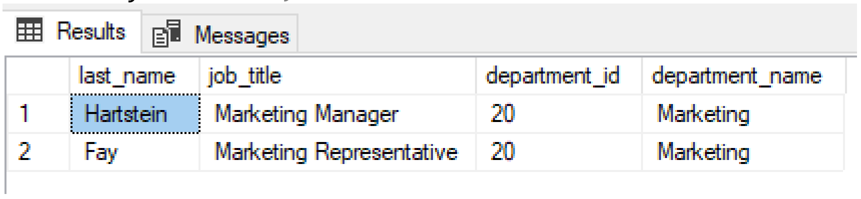
```
SELECT last_name, department_name
FROM employees JOIN departments
ON employees.department_id = departments.department_id;
```


Results | Messages

	last_name	department_name
1	King	Executive
2	Kochhar	Executive
3	De Haan	Executive
4	Hunold	IT
5	Ernst	IT
6	Austin	IT
7	Pataballa	IT
8	Lorentz	IT
9	Greenberg	Finance
10	Faviet	Finance
11	Chen	Finance
12	Sciarra	Finance
13	Urman	Finance
14	Popp	Finance
15	Raphaely	Purchasing
16	Khoo	Purchasing
17	Baida	Purchasing
18	Tobias	Purchasing
19	Himuro	Purchasing
20	Colmenares	Purchasing

17. 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';
```


Results | Messages

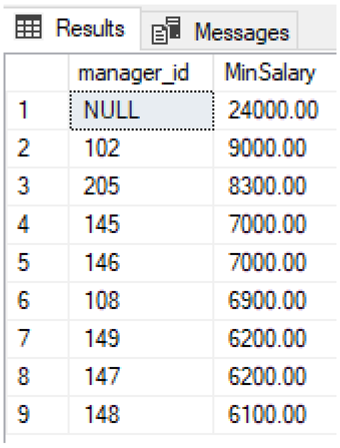
	last_name	job_title	department_id	department_name
1	Hartstein	Marketing Manager	20	Marketing
2	Fay	Marketing Representative	20	Marketing

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
```

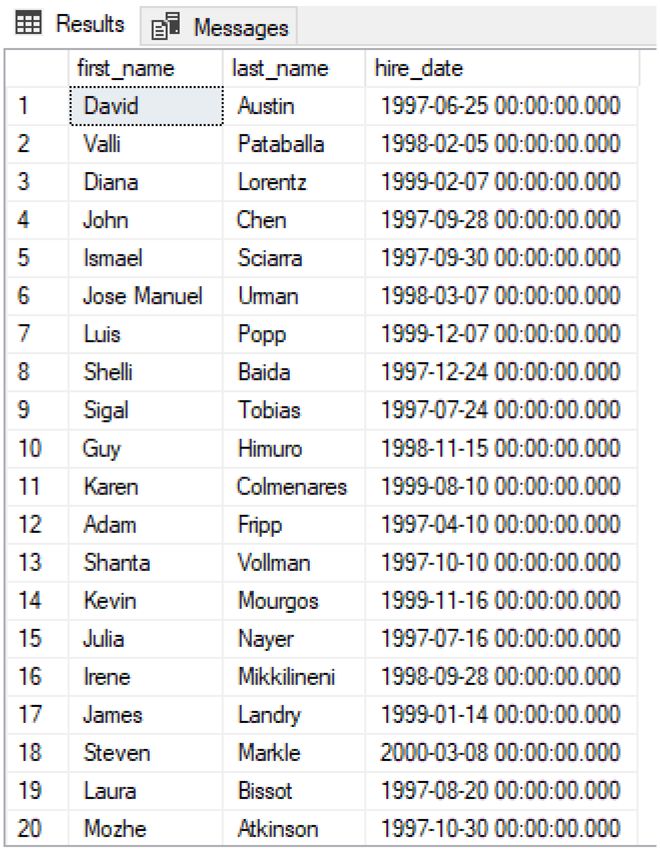
```
FROM employees
GROUP BY manager_id
HAVING MIN(salary) > 6000
ORDER BY MinSalary DESC;
```



	manager_id	MinSalary
1	NULL	24000.00
2	102	9000.00
3	205	8300.00
4	145	7000.00
5	146	7000.00
6	108	6900.00
7	149	6200.00
8	147	6200.00
9	148	6100.00

2. 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.

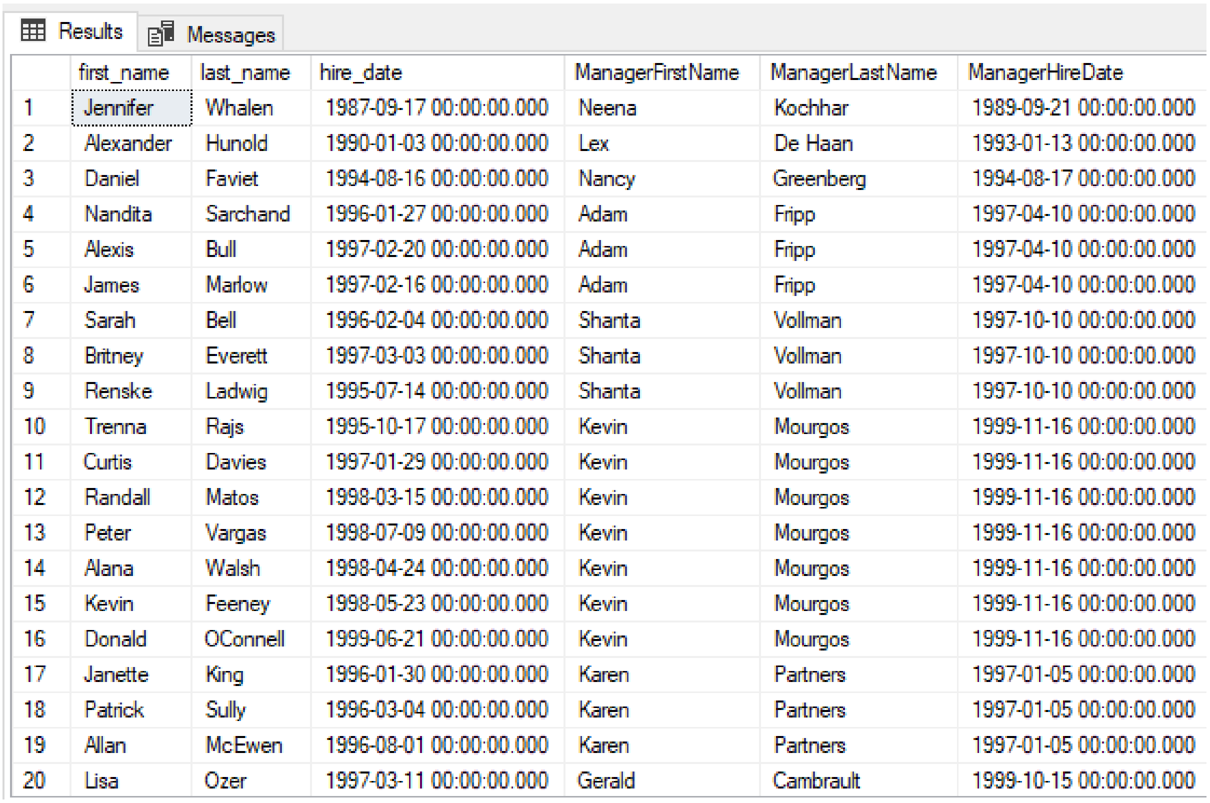
```
SELECT first_name, last_name, hire_date
FROM employees
WHERE hire_date > (
                  SELECT hire_date
                  FROM employees
                  WHERE last_name = 'Davies'
                  );
```



	first_name	last_name	hire_date
1	David	Austin	1997-06-25 00:00:00.000
2	Valli	Pataballa	1998-02-05 00:00:00.000
3	Diana	Lorentz	1999-02-07 00:00:00.000
4	John	Chen	1997-09-28 00:00:00.000
5	Ismael	Sciarra	1997-09-30 00:00:00.000
6	Jose Manuel	Urman	1998-03-07 00:00:00.000
7	Luis	Popp	1999-12-07 00:00:00.000
8	Shelli	Baida	1997-12-24 00:00:00.000
9	Sigal	Tobias	1997-07-24 00:00:00.000
10	Guy	Himuro	1998-11-15 00:00:00.000
11	Karen	Colmenares	1999-08-10 00:00:00.000
12	Adam	Fripp	1997-04-10 00:00:00.000
13	Shanta	Vollman	1997-10-10 00:00:00.000
14	Kevin	Mourgos	1999-11-16 00:00:00.000
15	Julia	Nayer	1997-07-16 00:00:00.000
16	Irene	Mikkilineni	1998-09-28 00:00:00.000
17	James	Landry	1999-01-14 00:00:00.000
18	Steven	Markle	2000-03-08 00:00:00.000
19	Laura	Bissot	1997-08-20 00:00:00.000
20	Mozhe	Atkinson	1997-10-30 00:00:00.000

3. 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.

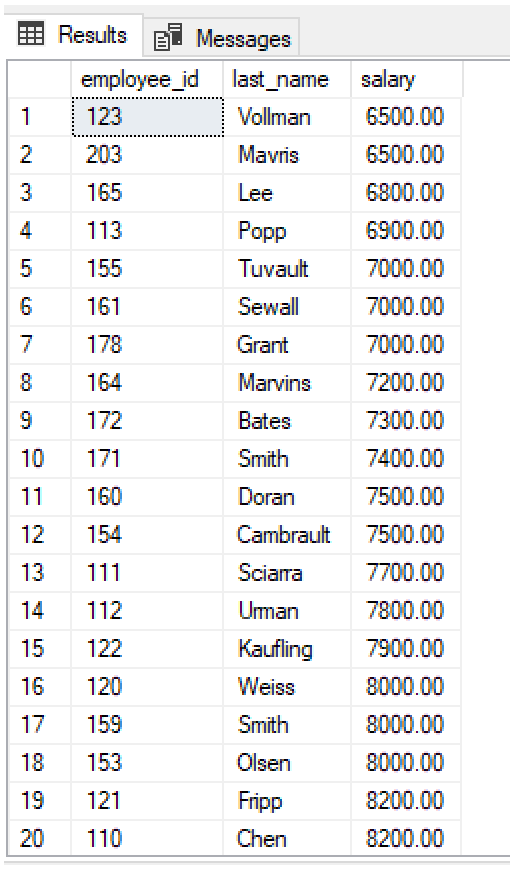
```
SELECT e1.first_name, e1.last_name, e1.hire_date, e2.first_name as ManagerFirstName,
e2.last_name as ManagerLastName, e2.hire_date as ManagerHireDate
FROM employees e1
       JOIN employees e2
       ON e1.manager_id = e2.employee_id
WHERE e1.hire_date < e2.hire_date;
```



	first_name	last_name	hire_date	ManagerFirstName	ManagerLastName	ManagerHireDate
1	Jennifer	Whalen	1987-09-17 00:00:00.000	Neena	Kochhar	1989-09-21 00:00:00.000
2	Alexander	Hunold	1990-01-03 00:00:00.000	Lex	De Haan	1993-01-13 00:00:00.000
3	Daniel	Faviet	1994-08-16 00:00:00.000	Nancy	Greenberg	1994-08-17 00:00:00.000
4	Nandita	Sarchand	1996-01-27 00:00:00.000	Adam	Fripp	1997-04-10 00:00:00.000
5	Alexis	Bull	1997-02-20 00:00:00.000	Adam	Fripp	1997-04-10 00:00:00.000
6	James	Marlow	1997-02-16 00:00:00.000	Adam	Fripp	1997-04-10 00:00:00.000
7	Sarah	Bell	1996-02-04 00:00:00.000	Shanta	Vollman	1997-10-10 00:00:00.000
8	Britney	Everett	1997-03-03 00:00:00.000	Shanta	Vollman	1997-10-10 00:00:00.000
9	Renske	Ladwig	1995-07-14 00:00:00.000	Shanta	Vollman	1997-10-10 00:00:00.000
10	Trenna	Rajs	1995-10-17 00:00:00.000	Kevin	Mourgos	1999-11-16 00:00:00.000
11	Curtis	Davies	1997-01-29 00:00:00.000	Kevin	Mourgos	1999-11-16 00:00:00.000
12	Randall	Matos	1998-03-15 00:00:00.000	Kevin	Mourgos	1999-11-16 00:00:00.000
13	Peter	Vargas	1998-07-09 00:00:00.000	Kevin	Mourgos	1999-11-16 00:00:00.000
14	Alana	Walsh	1998-04-24 00:00:00.000	Kevin	Mourgos	1999-11-16 00:00:00.000
15	Kevin	Feeney	1998-05-23 00:00:00.000	Kevin	Mourgos	1999-11-16 00:00:00.000
16	Donald	OConnell	1999-06-21 00:00:00.000	Kevin	Mourgos	1999-11-16 00:00:00.000
17	Janette	King	1996-01-30 00:00:00.000	Karen	Partners	1997-01-05 00:00:00.000
18	Patrick	Sully	1996-03-04 00:00:00.000	Karen	Partners	1997-01-05 00:00:00.000
19	Allan	McEwen	1996-08-01 00:00:00.000	Karen	Partners	1997-01-05 00:00:00.000
20	Lisa	Ozer	1997-03-11 00:00:00.000	Gerald	Cambrault	1999-10-15 00:00:00.000

4. 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.

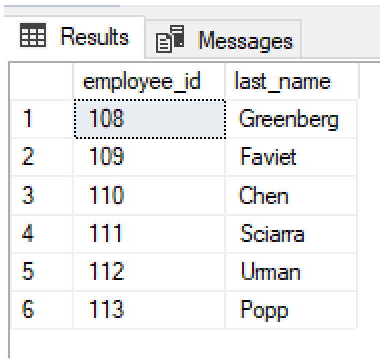
```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (
              SELECT AVG(salary)
              FROM employees
              )
ORDER BY salary;
```


Results | Messages

	employee_id	last_name	salary
1	123	Vollman	6500.00
2	203	Mavris	6500.00
3	165	Lee	6800.00
4	113	Popp	6900.00
5	155	Tuvault	7000.00
6	161	Sewall	7000.00
7	178	Grant	7000.00
8	164	Marvins	7200.00
9	172	Bates	7300.00
10	171	Smith	7400.00
11	160	Doran	7500.00
12	154	Cambrault	7500.00
13	111	Sciarra	7700.00
14	112	Urman	7800.00
15	122	Kaufling	7900.00
16	120	Weiss	8000.00
17	159	Smith	8000.00
18	153	Olsen	8000.00
19	121	Fripp	8200.00
20	110	Chen	8200.00

5. 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".

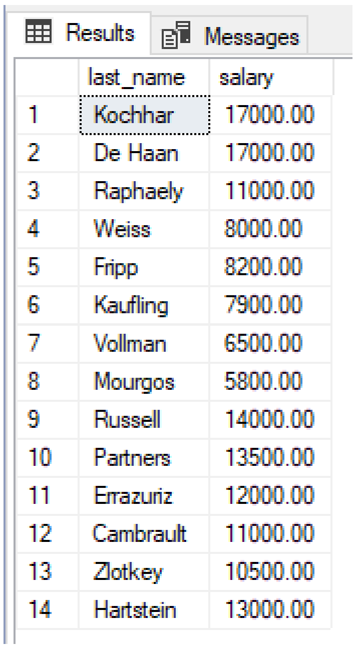
```
SELECT employee_id, last_name
FROM employees
WHERE department_id = (
                      SELECT department_id
                      FROM employees
                      WHERE last_name LIKE 'U%'
                      );
```


Results | Messages

	employee_id	last_name
1	108	Greenberg
2	109	Faviet
3	110	Chen
4	111	Sciarra
5	112	Urman
6	113	Popp

6. 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';
```

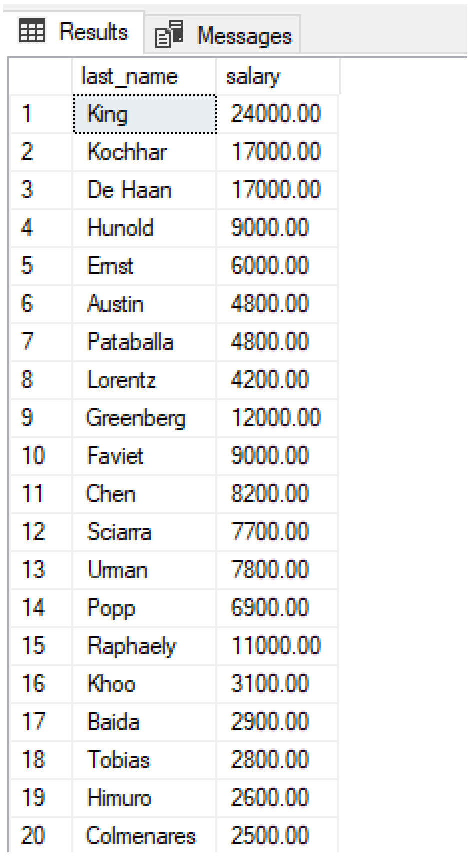


Results | Messages

	last_name	salary
1	Kochhar	17000.00
2	De Haan	17000.00
3	Raphaely	11000.00
4	Weiss	8000.00
5	Fripp	8200.00
6	Kaufling	7900.00
7	Vollman	6500.00
8	Mourgos	5800.00
9	Russell	14000.00
10	Partners	13500.00
11	Errazuriz	12000.00
12	Cambrault	11000.00
13	Zlotkey	10500.00
14	Hartstein	13000.00

7. 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;
```



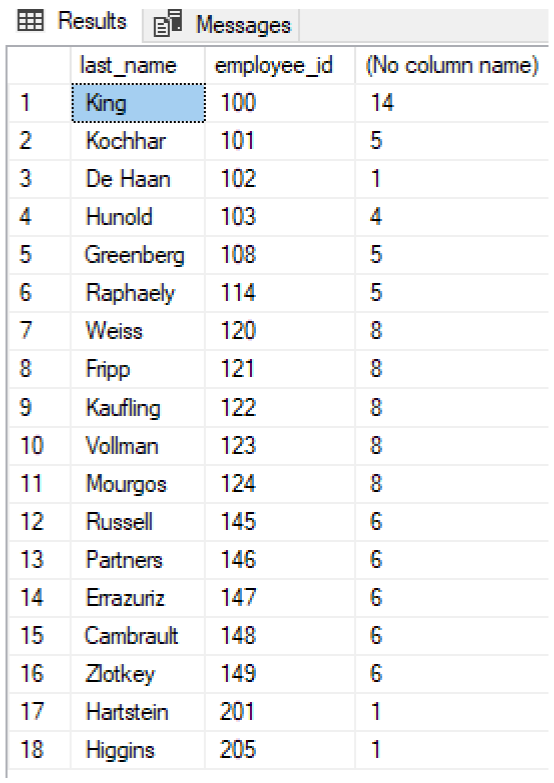
Results | Messages

	last_name	salary
1	King	24000.00
2	Kochhar	17000.00
3	De Haan	17000.00
4	Hunold	9000.00
5	Ernst	6000.00
6	Austin	4800.00
7	Pataballa	4800.00
8	Lorentz	4200.00
9	Greenberg	12000.00
10	Faviet	9000.00
11	Chen	8200.00
12	Sciarra	7700.00
13	Urman	7800.00
14	Popp	6900.00
15	Raphaely	11000.00
16	Khoo	3100.00
17	Baida	2900.00
18	Tobias	2800.00
19	Himuro	2600.00
20	Colmenares	2500.00

TASK 4

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

```sql
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
```

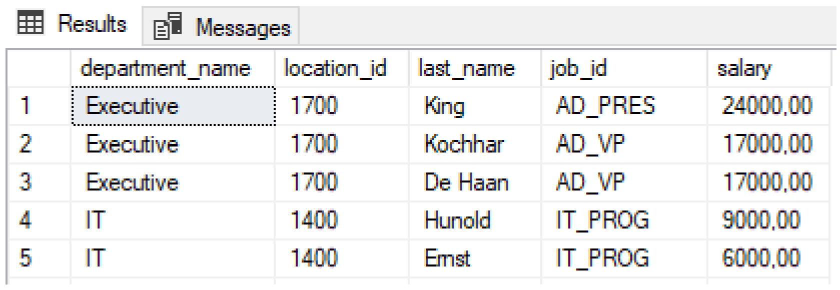


Results | Messages

	last_name	employee_id	(No column name)
1	King	100	14
2	Kochhar	101	5
3	De Haan	102	1
4	Hunold	103	4
5	Greenberg	108	5
6	Raphaely	114	5
7	Weiss	120	8
8	Fripp	121	8
9	Kaufling	122	8
10	Vollman	123	8
11	Mourgos	124	8
12	Russell	145	6
13	Partners	146	6
14	Errazuriz	147	6
15	Cambrault	148	6
16	Zlotkey	149	6
17	Hartstein	201	1
18	Higgins	205	1

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

```sql
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
```

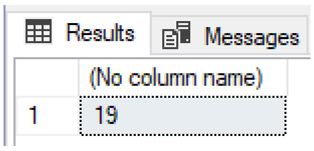


Results | Messages

	department_name	location_id	last_name	job_id	salary
1	Executive	1700	King	AD_PRES	24000,00
2	Executive	1700	Kochhar	AD_VP	17000,00
3	Executive	1700	De Haan	AD_VP	17000,00
4	IT	1400	Hunold	IT_PROG	9000,00
5	IT	1400	Ernst	IT_PROG	6000,00

3. 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'
```

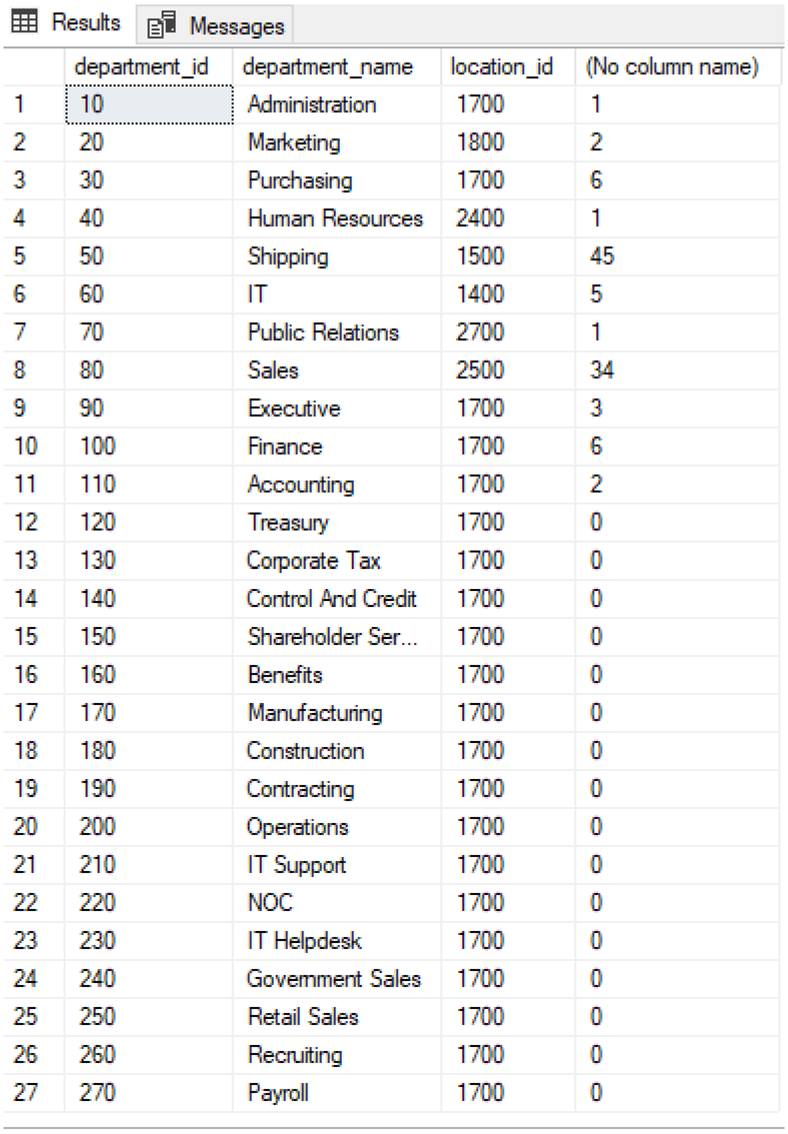


Results | Messages

	(No column name)
1	19

4. 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
```

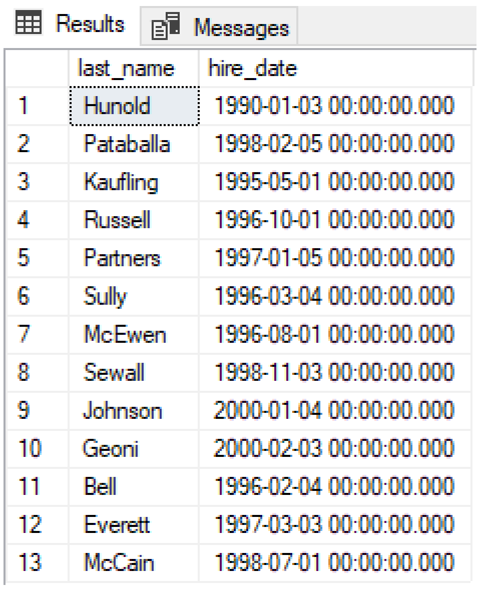


Results | Messages

	department_id	department_name	location_id	(No column name)
1	10	Administration	1700	1
2	20	Marketing	1800	2
3	30	Purchasing	1700	6
4	40	Human Resources	2400	1
5	50	Shipping	1500	45
6	60	IT	1400	5
7	70	Public Relations	2700	1
8	80	Sales	2500	34
9	90	Executive	1700	3
10	100	Finance	1700	6
11	110	Accounting	1700	2
12	120	Treasury	1700	0
13	130	Corporate Tax	1700	0
14	140	Control And Credit	1700	0
15	150	Shareholder Ser...	1700	0
16	160	Benefits	1700	0
17	170	Manufacturing	1700	0
18	180	Construction	1700	0
19	190	Contracting	1700	0
20	200	Operations	1700	0
21	210	IT Support	1700	0
22	220	NOC	1700	0
23	230	IT Helpdesk	1700	0
24	240	Government Sales	1700	0
25	250	Retail Sales	1700	0
26	260	Recruiting	1700	0
27	270	Payroll	1700	0

5. 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
```

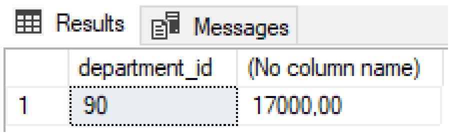


Results | Messages

	last_name	hire_date
1	Hunold	1990-01-03 00:00:00.000
2	Pataballa	1998-02-05 00:00:00.000
3	Kaufling	1995-05-01 00:00:00.000
4	Russell	1996-10-01 00:00:00.000
5	Partners	1997-01-05 00:00:00.000
6	Sully	1996-03-04 00:00:00.000
7	McEwen	1996-08-01 00:00:00.000
8	Sewall	1998-11-03 00:00:00.000
9	Johnson	2000-01-04 00:00:00.000
10	Geoni	2000-02-03 00:00:00.000
11	Bell	1996-02-04 00:00:00.000
12	Everett	1997-03-03 00:00:00.000
13	McCain	1998-07-01 00:00:00.000

6. 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
```

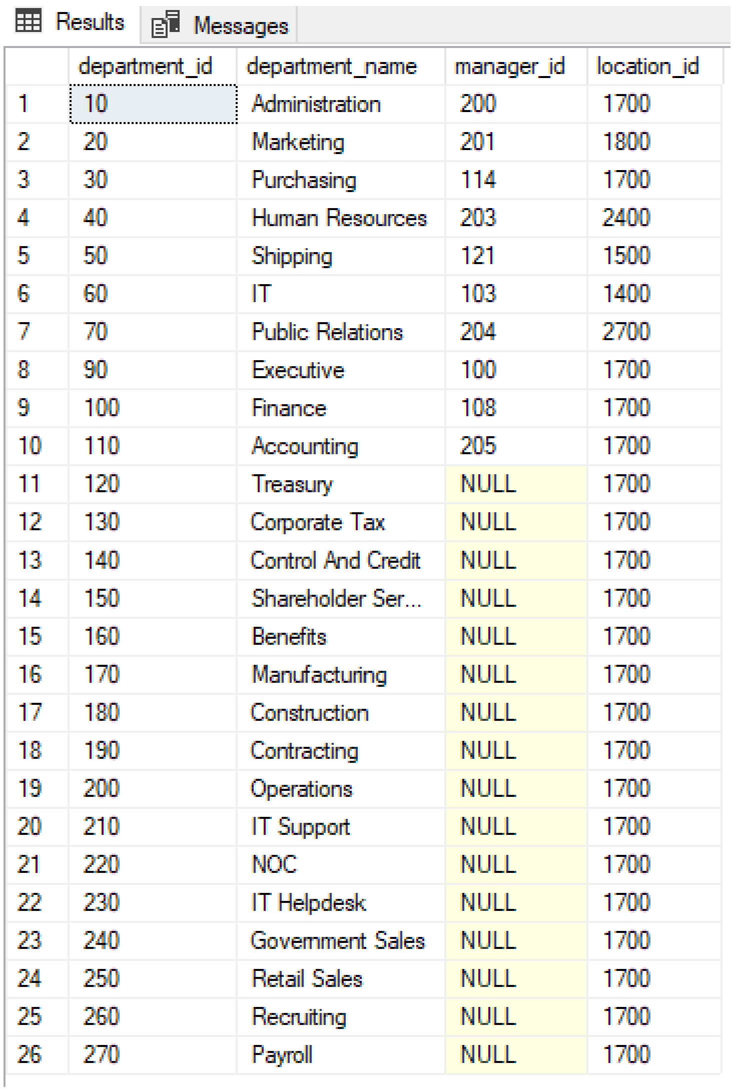


Results | Messages

	department_id	(No column name)
1	90	17000.00

7. 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);
```

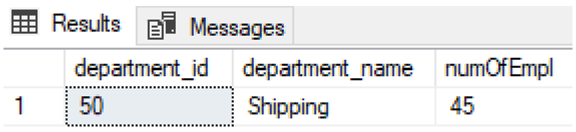

Results Messages

	department_id	department_name	manager_id	location_id
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	90	Executive	100	1700
9	100	Finance	108	1700
10	110	Accounting	205	1700
11	120	Treasury	NULL	1700
12	130	Corporate Tax	NULL	1700
13	140	Control And Credit	NULL	1700
14	150	Shareholder Ser...	NULL	1700
15	160	Benefits	NULL	1700
16	170	Manufacturing	NULL	1700
17	180	Construction	NULL	1700
18	190	Contracting	NULL	1700
19	200	Operations	NULL	1700
20	210	IT Support	NULL	1700
21	220	NOC	NULL	1700
22	230	IT Helpdesk	NULL	1700
23	240	Government Sales	NULL	1700
24	250	Retail Sales	NULL	1700
25	260	Recruiting	NULL	1700
26	270	Payroll	NULL	1700

8. Display the department number, department name and the number of employees for the department:

a. with the highest number of employees.

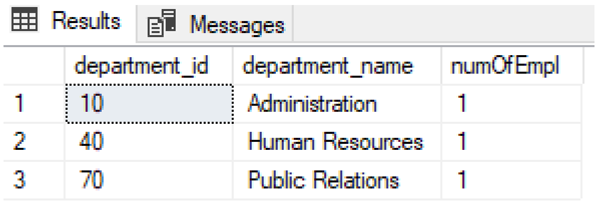
```
WITH emplTable (department_id, department_name, numOfEmpl)
AS (
       SELECT d1.department_id, d1.department_name, count(e1.last_name) AS numOfEmpl
       FROM departments d1
       JOIN employees e1 ON d1.department_id = e1.department_id
       GROUP BY d1.department_id, d1.department_name
       )
SELECT department_id, department_name, numOfEmpl
FROM emplTable
WHERE numOfEmpl = (
              SELECT MAX(numOfEmpl)
              FROM emplTable
              );
```


Results Messages

	department_id	department_name	numOfEmpl
1	50	Shipping	45

b. with the lowest number of employees

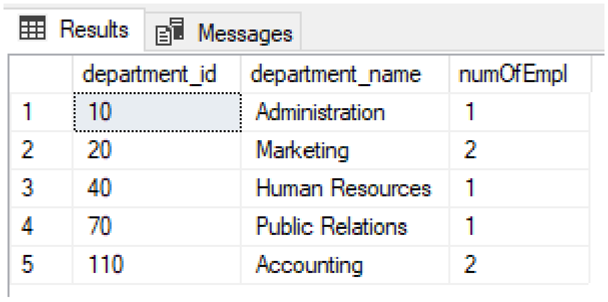
```
WITH emplTable (department_id, department_name, numOfEmpl)
AS (
       SELECT d1.department_id, d1.department_name, count(e1.last_name) AS numOfEmpl
       FROM departments d1
       JOIN employees e1 ON d1.department_id = e1.department_id
       GROUP BY d1.department_id, d1.department_name
       )
SELECT department_id, department_name, numOfEmpl
FROM emplTable
WHERE numOfEmpl = (
              SELECT MIN(numOfEmpl)
              FROM emplTable
              );
```


Results | Messages

	department_id	department_name	numOfEmpl
1	10	Administration	1
2	40	Human Resources	1
3	70	Public Relations	1

c. that employs fewer than three employees.

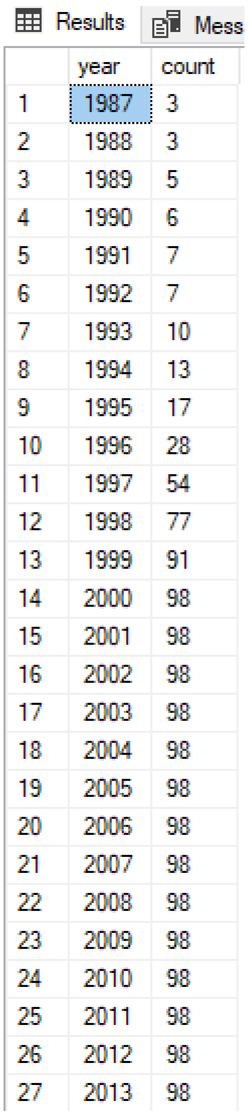
```
WITH emplTable (department_id, department_name, numOfEmpl)
AS (
       SELECT d1.department_id, d1.department_name, count(e1.last_name) AS numOfEmpl
       FROM departments d1
       JOIN employees e1 ON d1.department_id = e1.department_id
       GROUP BY d1.department_id, d1.department_name
       )
SELECT department_id, department_name, numOfEmpl
FROM emplTable
WHERE numOfEmpl < 3;
```

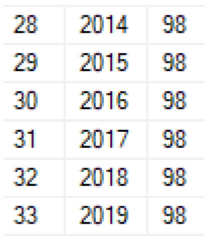

Results | Messages

	department_id	department_name	numOfEmpl
1	10	Administration	1
2	20	Marketing	2
3	40	Human Resources	1
4	70	Public Relations	1
5	110	Accounting	2

9. 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
```

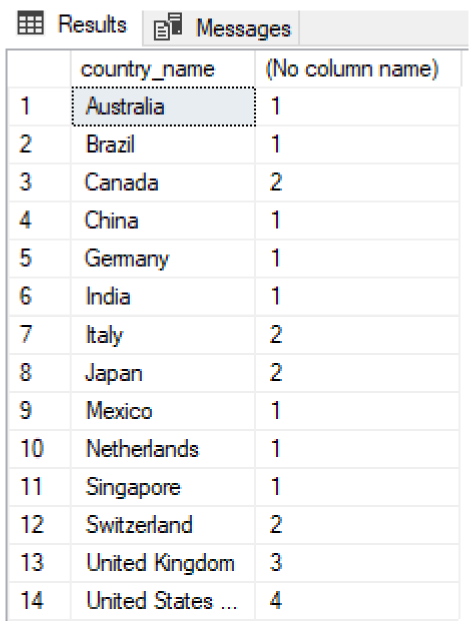




	year	count
1	1987	3
2	1988	3
3	1989	5
4	1990	6
5	1991	7
6	1992	7
7	1993	10
8	1994	13
9	1995	17
10	1996	28
11	1997	54
12	1998	77
13	1999	91
14	2000	98
15	2001	98
16	2002	98
17	2003	98
18	2004	98
19	2005	98
20	2006	98
21	2007	98
22	2008	98
23	2009	98
24	2010	98
25	2011	98
26	2012	98
27	2013	98
28	2014	98
29	2015	98
30	2016	98
31	2017	98
32	2018	98
33	2019	98

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

```sql
SELECT country_name, count(l1.country_id)
FROM countries c1 join locations l1
ON c1.country_id = l1.country_id
GROUP BY country_name
```

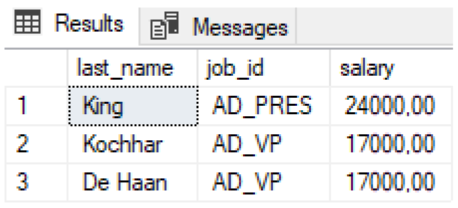


	country_name	(No column name)
1	Australia	1
2	Brazil	1
3	Canada	2
4	China	1
5	Germany	1
6	India	1
7	Italy	2
8	Japan	2
9	Mexico	1
10	Netherlands	1
11	Singapore	1
12	Switzerland	2
13	United Kingdom	3
14	United States ...	4

Additional exercises

1A. 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.

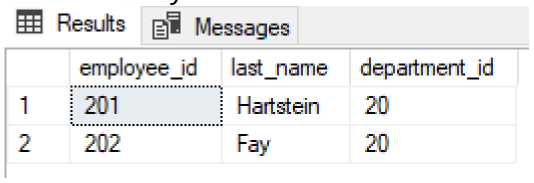
```
SELECT last_name, job_id, salary
FROM employees
WHERE salary > (SELECT MAX(salary)
                FROM employees
                       WHERE job_id = 'SA_MAN');
```



	last_name	job_id	salary
1	King	AD_PRES	24000,00
2	Kochhar	AD_VP	17000,00
3	De Haan	AD_VP	17000,00

2A. 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'.

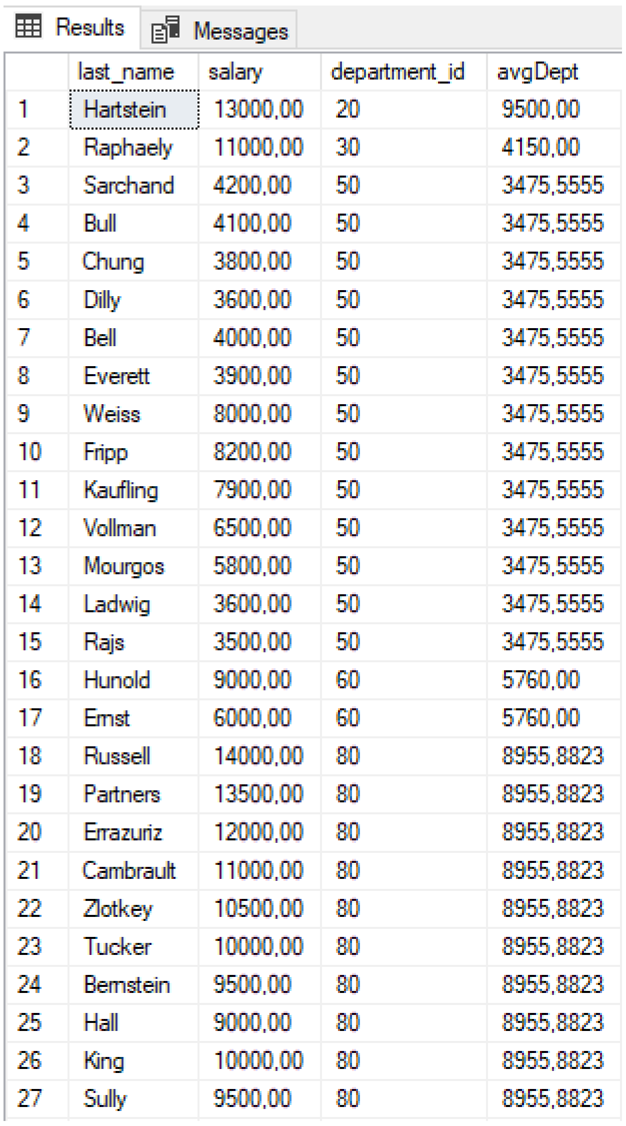
```
SELECT employee_id, last_name, e1.department_id
FROM employees e1
JOIN (
       departments d1 JOIN locations l1 ON d1.location_id = l1.location_id
       ) ON e1.department_id = d1.department_id
WHERE l1.city LIKE 'T%'
```

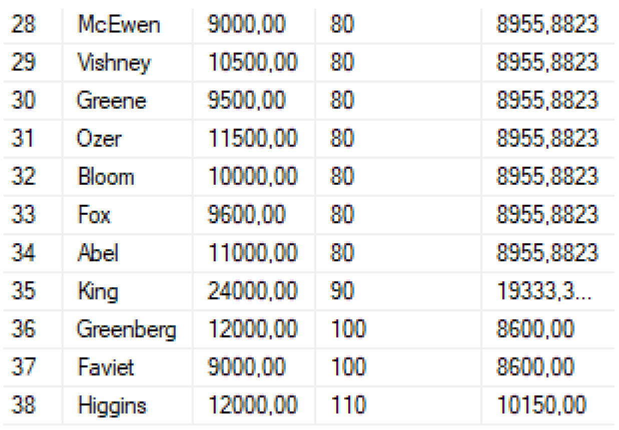


	employee_id	last_name	department_id
1	201	Hartstein	20
2	202	Fay	20

3A. Write a query to find all employees who earn more than the average salary in their department.

```sql
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
```

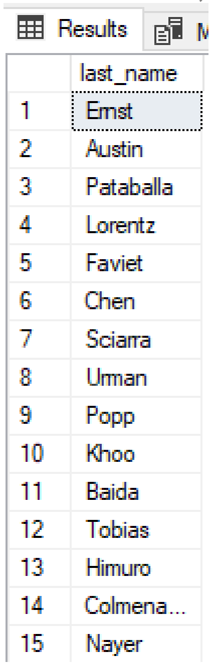




	last_name	salary	department_id	avgDept
1	Hartstein	13000,00	20	9500,00
2	Raphaely	11000,00	30	4150,00
3	Sarchand	4200,00	50	3475,5555
4	Bull	4100,00	50	3475,5555
5	Chung	3800,00	50	3475,5555
6	Dilly	3600,00	50	3475,5555
7	Bell	4000,00	50	3475,5555
8	Everett	3900,00	50	3475,5555
9	Weiss	8000,00	50	3475,5555
10	Fripp	8200,00	50	3475,5555
11	Kaufling	7900,00	50	3475,5555
12	Vollman	6500,00	50	3475,5555
13	Mourgos	5800,00	50	3475,5555
14	Ladwig	3600,00	50	3475,5555
15	Rajs	3500,00	50	3475,5555
16	Hunold	9000,00	60	5760,00
17	Ernst	6000,00	60	5760,00
18	Russell	14000,00	80	8955,8823
19	Partners	13500,00	80	8955,8823
20	Errazuriz	12000,00	80	8955,8823
21	Cambrault	11000,00	80	8955,8823
22	Zlotkey	10500,00	80	8955,8823
23	Tucker	10000,00	80	8955,8823
24	Bernstein	9500,00	80	8955,8823
25	Hall	9000,00	80	8955,8823
26	King	10000,00	80	8955,8823
27	Sully	9500,00	80	8955,8823
28	McEwen	9000,00	80	8955,8823
29	Vishney	10500,00	80	8955,8823
30	Greene	9500,00	80	8955,8823
31	Ozer	11500,00	80	8955,8823
32	Bloom	10000,00	80	8955,8823
33	Fox	9600,00	80	8955,8823
34	Abel	11000,00	80	8955,8823
35	King	24000,00	90	19333,3...
36	Greenberg	12000,00	100	8600,00
37	Faviet	9000,00	100	8600,00
38	Higgins	12000,00	110	10150,00

4A. Find all employees who are not supervisors (managers). Do this using the NOT EXISTS operator.

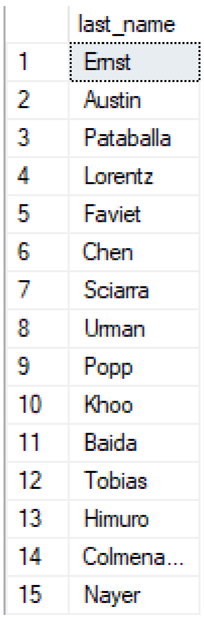
```
SELECT EMPL.last_name
FROM employees EMPL
WHERE NOT EXISTS (
              SELECT MGRS_TUPLE_LIST.employee_id
              FROM employees MGRS_TUPLE_LIST
              JOIN (
                     SELECT e1.manager_id
                     FROM employees e1
                     GROUP BY e1.manager_id
                     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
              );
```


Results

	last_name
1	Ernst
2	Austin
3	Pataballa
4	Lorentz
5	Faviet
6	Chen
7	Sciarra
8	Urman
9	Popp
10	Khoo
11	Baida
12	Tobias
13	Himuro
14	Colmena...
15	Nayer

Can it be done using NOT IN?

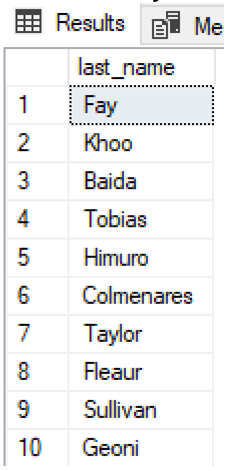
```
SELECT EMPL.last_name
FROM employees EMPL
WHERE EMPL.employee_id NOT IN (
              SELECT MGRS_TUPLE_LIST.employee_id
              FROM employees MGRS_TUPLE_LIST
              JOIN (
                     SELECT e1.manager_id
                     FROM employees e1
                     GROUP 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
              );
```



	last_name
1	Ernst
2	Austin
3	Pataballa
4	Lorentz
5	Faviet
6	Chen
7	Sciarra
8	Urman
9	Popp
10	Khoo
11	Baida
12	Tobias
13	Himuro
14	Colmena...
15	Nayer

5A. Display the last names of the employees who earn less than the average salary in their departments.

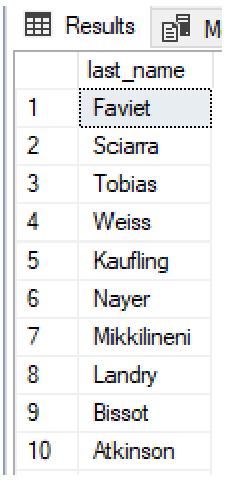
```
SELECT last_name
FROM employees e1
JOIN (
       SELECT department_id, AVG(salary) AS avg_dept
       FROM employees
       GROUP BY department_id
       ) AS avg_dept_list ON e1.department_id = avg_dept_list.department_id
WHERE salary < avg_dept
```



	last_name
1	Fay
2	Khoo
3	Baida
4	Tobias
5	Himuro
6	Colmenares
7	Taylor
8	Fleaur
9	Sullivan
10	Geoni

6A. Display the last names of the employees who have one or more co-workers in their departments with later hire dates but higher salaries.

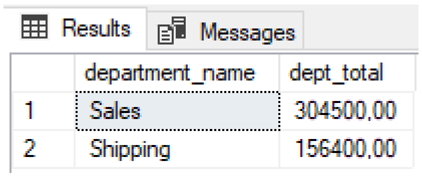
```
SELECT last_name
FROM (
       SELECT DISTINCT e1.last_name, e1.employee_id
       FROM employees e1
       JOIN employees e2 ON e1.department_id = e2.department_id
       WHERE e2.hire_date > e1.hire_date
              AND e2.salary > e1.salary
       ) AS emplCol;
```


Results | M

	last_name
1	Faviet
2	Sciarra
3	Tobias
4	Weiss
5	Kaufling
6	Nayer
7	Mikkilineni
8	Landry
9	Bissot
10	Atkinson

7A. 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.

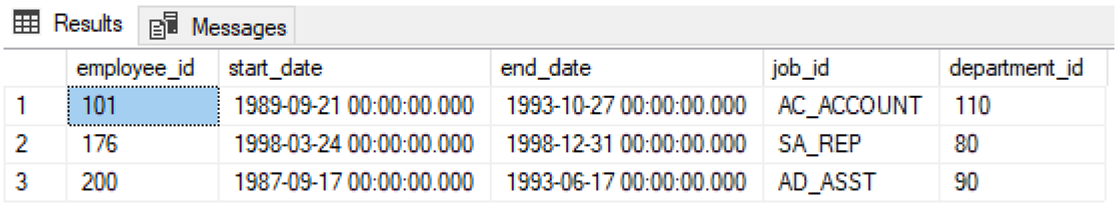
```
WITH Summary
AS (
       SELECT d1.department_name, SUM(e1.salary) AS dept_total
       FROM departments d1
       JOIN employees e1 ON d1.department_id = e1.department_id
       GROUP BY d1.department_name
       )
SELECT department_name, dept_total
FROM Summary
WHERE dept_total > (
              SELECT SUM(salary) / 8
              FROM employees
              )
```


Results | Messages

	department_name	dept_total
1	Sales	304500,00
2	Shipping	156400,00

8A. 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
                     )
```



	employee_id	start_date	end_date	job_id	department_id
1	101	1989-09-21 00:00:00.000	1993-10-27 00:00:00.000	AC_ACCOUNT	110
2	176	1998-03-24 00:00:00.000	1998-12-31 00:00:00.000	SA_REP	80
3	200	1987-09-17 00:00:00.000	1993-06-17 00:00:00.000	AD_ASST	90