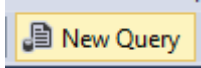


# Task 1

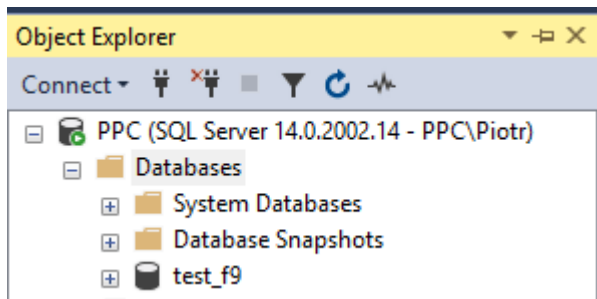
1. Choose New Query option for opening SQL worksheet window.



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, primary 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: member\_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	Ropebum	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;

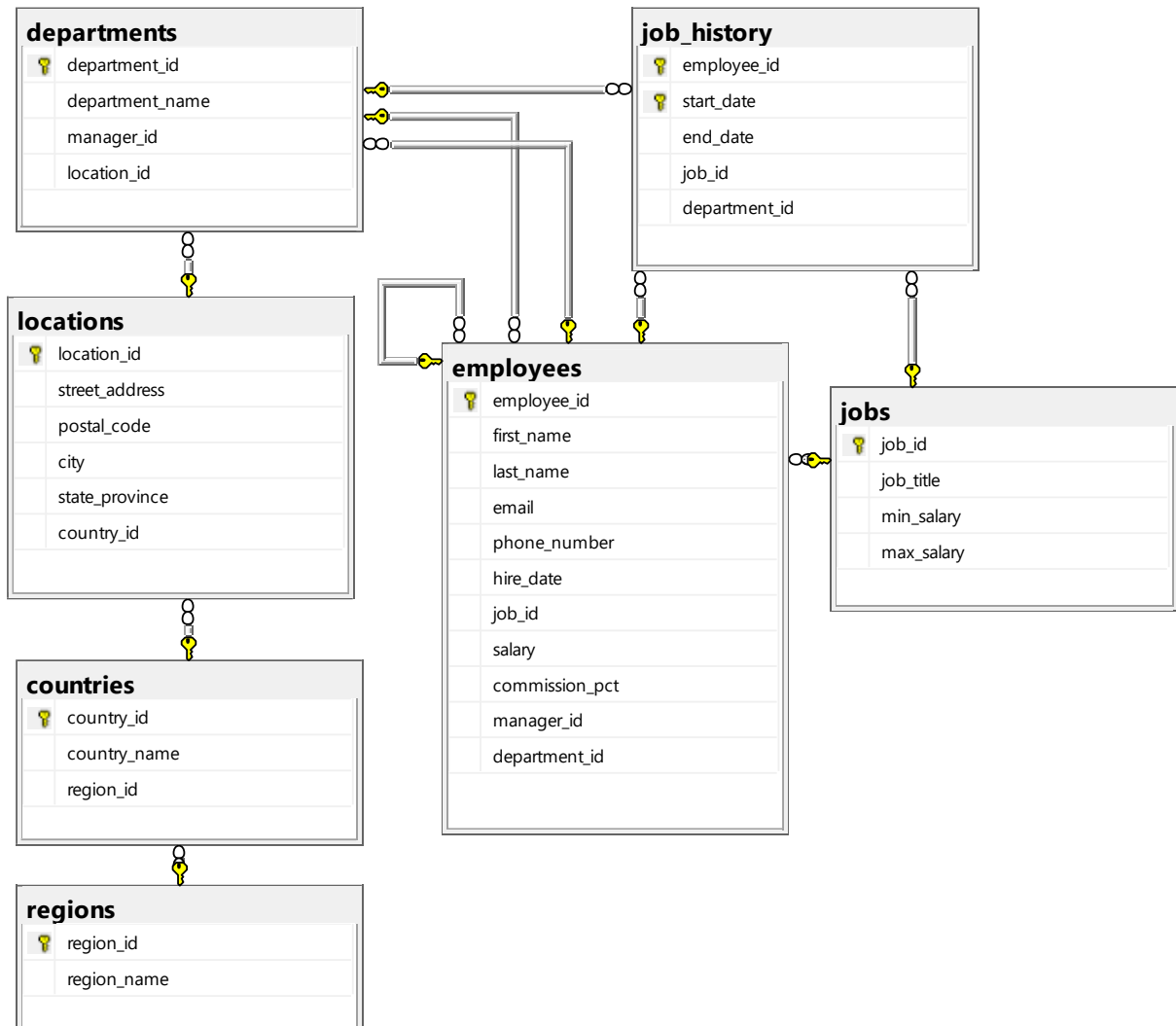
	BOOK_DATE	COPY_ID	MEMBER_ID	TITLE_ID	ACT_RET_DATE	EXP_RET_DATE
1	2019-05-25	1	1	1	2019-05-28 11:17:16.470	2019-05-29 11:17:16.470
2	2019-05-26	1	6	6	2019-05-28 11:17:16.470	2019-05-28 11:17:16.470
3	2019-05-27	1	3	7	NULL	2019-05-29 11:17:16.470
4	2019-05-28	3	2	4	NULL	2019-05-30 11:17:16.470
5	2019-05-29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

```
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

20	Karen	Colmenares	2500.00
21	Matthew	Weiss	8000.00
22	Adam	Fripp	8200.00
23	Payam	Kaufling	7900.00
24	Shanta	Vollman	6500.00
25	Kevin	Mourgos	5800.00
26	Julia	Nayer	3200.00
27	Irene	Mikkilineni	2700.00
28	James	Landry	2400.00
29	Steven	Markle	2200.00
30	Laura	Bissot	3300.00
31	Mozhe	Atkinson	2800.00
32	James	Marlow	2500.00
33	TJ	Olson	2100.00
34	Jason	Mallin	3300.00
35	Michael	Rogers	2900.00
36	Ki	Gee	2400.00
37	Hazel	Philtanker	2200.00
38	Renske	Ladwig	3600.00
39	Stephen	Stiles	3200.00
40	John	Seo	2700.00

40	John	Seo	2700.00
41	Joshua	Patel	2500.00
42	Trenna	Rajs	3500.00
43	Curtis	Davies	3100.00
44	Randall	Matos	2600.00
45	Peter	Vargas	2500.00
46	John	Russell	14000.00
47	Karen	Partners	13500.00
48	Alberto	Errazuriz	12000.00
49	Gerald	Cambrault	11000.00
50	Eleni	Zlotkey	10500.00
51	Peter	Tucker	10000.00
52	David	Bernstein	9500.00
53	Peter	Hall	9000.00
54	Christopher	Olsen	8000.00
55	Nanette	Cambrault	7500.00
56	Oliver	Tuvault	7000.00
57	Janette	King	10000.00
58	Patrick	Sully	9500.00
59	Allan	McEwen	9000.00
60	Lindsey	Smith	8000.00

61	Louise	Doran	7500.00
62	Sarath	Sewall	7000.00
63	Clara	Vishney	10500.00
64	Danielle	Greene	9500.00
65	Mattea	Marvins	7200.00
66	David	Lee	6800.00
67	Sundar	Ande	6400.00
68	Amit	Banda	6200.00
69	Lisa	Ozer	11500.00
70	Hamison	Bloom	10000.00
71	Tayler	Fox	9600.00
72	William	Smith	7400.00
73	Elizabeth	Bates	7300.00
74	Sundita	Kumar	6100.00
75	Ellen	Abel	11000.00
76	Alyssa	Hutton	8800.00
77	Jonathon	Taylor	8600.00
78	Jack	Livingston	8400.00
79	Kimberely	Grant	7000.00
80	Charles	Johnson	6200.00

81	Winston	Taylor	3200.00				
82	Jean	Fleur	3100.00				
83	Martha	Sullivan	2500.00				
84	Girard	Geoni	2800.00				
85	Nandita	Sarchand	4200.00				
86	Alexis	Bull	4100.00				
87	Julia	Dellinger	3400.00				
88	Anthony	Cabrio	3000.00				
89	Kelly	Chung	3800.00				
90	Jennifer	Dilly	3600.00				
91	Timothy	Gates	2900.00				
92	Randall	Perkins	2500.00				
93	Sarah	Bell	4000.00	100	Douglas	Grant	2600.00
94	Britney	Everett	3900.00	101	Jennifer	Whalen	4400.00
95	Samuel	McCain	3200.00	102	Michael	Hartstein	13000.00
96	Vance	Jones	2800.00	103	Pat	Fay	6000.00
97	Alana	Walsh	3100.00	104	Susan	Mavris	6500.00
98	Kevin	Feeney	3000.00	105	Hermann	Baer	10000.00
99	Donald	OConnell	2600.00	106	Shelley	Higgins	12000.00
100	Douglas	Grant	2600.00	107	William	Gietz	8300.00

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

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

Results		Messages
	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;
```

Results		Messages
	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	21	Rogers	2900.00
2	Kochhar	17000.00	22	Gee	2400.00
3	De Haan	17000.00	23	Philtanker	2200.00
4	Austin	4800.00	24	Ladwig	3600.00
5	Pataballa	4800.00	25	Stiles	3200.00
6	Lorentz	4200.00	26	Seo	2700.00
7	Khoo	3100.00	27	Patel	2500.00
8	Baida	2900.00	28	Rajs	3500.00
9	Tobias	2800.00	29	Davies	3100.00
10	Himuro	2600.00	30	Matos	2600.00
11	Colmenares	2500.00	31	Vargas	2500.00
12	Nayer	3200.00	32	Russell	14000.00
13	Mikkilineni	2700.00	33	Partners	13500.00
14	Landry	2400.00	34	Taylor	3200.00
15	Markle	2200.00	35	Fleaur	3100.00
16	Bissot	3300.00	36	Sullivan	2500.00
17	Atkinson	2800.00	37	Geoni	2800.00
18	Marlow	2500.00	38	Sarchand	4200.00
19	Olson	2100.00	39	Bull	4100.00
20	Mallin	3300.00	40	Dellinger	3400.00
			41	Cabrio	3000.00
			42	Chung	3800.00
			43	Dilly	3600.00
			44	Gates	2900.00
			45	Perkins	2500.00
			46	Bell	4000.00
			47	Everett	3900.00
			48	McCain	3200.00
			49	Jones	2800.00
			50	Walsh	3100.00
			51	Feeney	3000.00
			52	OConnell	2600.00
			53	Grant	2600.00
			54	Whalen	4400.00
			55	Hartstein	13000.00

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;
```

Results			Messages		
	last_name	department_id			
1	Atkinson	50	21	Kaufling	50
2	Bell	50	22	Ladwig	50
3	Bissot	50	23	Landry	50
4	Bull	50	24	Mallin	50
5	Cabrio	50	25	Markle	50
6	Chung	50	26	Marlow	50
7	Davies	50	27	Matos	50
8	Dellinger	50	28	McCain	50
9	Dilly	50	29	Mikkilineni	50
10	Everett	50	30	Mourgos	50
11	Fay	20	31	Nayer	50
12	Feeney	50	32	OConnell	50
13	Fleur	50	33	Olson	50
14	Fripp	50	34	Patel	50
15	Gates	50	35	Perkins	50
16	Gee	50	36	Philtanker	50
17	Geoni	50	37	Rajs	50
18	Grant	50	38	Rogers	50
19	Hartstein	20	39	Sarchand	50
20	Jones	50	40	Seo	50
			41	Stiles	50
			42	Sullivan	50
			43	Taylor	50
			44	Vargas	50
			45	Vollman	50
			46	Walsh	50
			47	Weiss	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

21	Smith	8000.00	0.30
22	Olsen	8000.00	0.20
23	Doran	7500.00	0.30
24	Cambrault	7500.00	0.20
25	Smith	7400.00	0.15
26	Bates	7300.00	0.15
27	Marvins	7200.00	0.10
28	Sewall	7000.00	0.25
29	Tuvault	7000.00	0.15
30	Grant	7000.00	0.15
31	Lee	6800.00	0.10
32	Ande	6400.00	0.10
33	Banda	6200.00	0.10
34	Johnson	6200.00	0.10
35	Kumar	6100.00	0.10

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 job_id, count(job_id)
FROM employees
GROUP BY job_id;
```

Results		Messages
	job_id	(No column name)
1	AC_ACCOUNT	1
2	AC_MGR	1
3	AD_ASST	1
4	AD PRES	1
5	AD_VP	2
6	FI_ACCOUNT	5
7	FI_MGR	1
8	HR_REP	1
9	IT_PROG	5
10	MK_MAN	1
11	MK_REP	1
12	PR_REP	1
13	PU_CLERK	5
14	PU_MAN	1
15	SA_MAN	5
16	SA_REP	30
17	SH_CLERK	20
18	ST_CLERK	20
19	ST_MAN	5

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;
```

Results Messages	
(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.

```
SELECT location_id, street_address, city, state_province, country_name
FROM locations JOIN countries
ON locations.country_id = countries.country_id;
```

Results Messages					
	location_id	street_address	city	state_province	country_name
1	1000	1297 Via Cola di Rie	Roma	NULL	Italy
2	1100	93091 Calle della Testa	Venice	NULL	Italy
3	1200	2017 Shinjuku-ku	Tokyo	Tokyo Prefecture	Japan
4	1300	9450 Kamiya-cho	Hiroshima	NULL	Japan
5	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
6	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
7	1600	2007 Zagora St	South Brunswick	New Jersey	United States of America
8	1700	2004 Charade Rd	Seattle	Washington	United States of America
9	1800	147 Spadina Ave	Toronto	Ontario	Canada
10	1900	6092 Boxwood St	Whitehorse	Yukon	Canada
11	2000	40-5-12 Laogianggen	Beijing	NULL	China
12	2100	1298 Vileparle (E)	Bombay	Maharashtra	India
13	2200	12-98 Victoria Street	Sydney	New South Wales	Australia
14	2300	198 Clementi North	Singapore	NULL	Singapore
15	2400	8204 Arthur St	London	NULL	United Kingdom
16	2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom
17	2600	9702 Chester Road	Stretford	Manchester	United Kingdom
18	2700	Schwanthalerstr. 7031	Munich	Bavaria	Germany
19	2800	Rua Frei Caneca 1360	Sao Paulo	Sao Paulo	Brazil
20	2900	20 Rue des Corps-Saints	Geneva	Geneve	Switzerland
21	3000	Murtenstrasse 921	Bern	BE	Switzerland
22	3100	Pieter Breughelstraat 837	Utrecht	Utrecht	Netherlands
23	3200	Mariano Escobedo 9991	Mexico City	Distrito Federal,	Mexico

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	Uman	Finance
14	Popp	Finance
15	Raphaely	Purchasing
16	Khoo	Purchasing
17	Baida	Purchasing
18	Tobias	Purchasing
19	Himuro	Purchasing
20	Colmenares	Purchasing

21	Weiss	Shipping
22	Fripp	Shipping
23	Kaufling	Shipping
24	Vollman	Shipping
25	Mourgos	Shipping
26	Nayer	Shipping
27	Mikkilineni	Shipping
28	Landry	Shipping
29	Markle	Shipping
30	Bissot	Shipping
31	Atkinson	Shipping
32	Marlow	Shipping
33	Olson	Shipping
34	Mallin	Shipping
35	Rogers	Shipping
36	Gee	Shipping
37	Philtanker	Shipping
38	Ladwig	Shipping
39	Stiles	Shipping
40	Seo	Shipping

41	Patel	Shipping
42	Rajs	Shipping
43	Davies	Shipping
44	Matos	Shipping
45	Vargas	Shipping
46	Russell	Sales
47	Partners	Sales
48	Errazuriz	Sales
49	Cambrault	Sales
50	Zlotkey	Sales
51	Tucker	Sales
52	Bernstein	Sales
53	Hall	Sales
54	Olsen	Sales
55	Cambrault	Sales
56	Tuvault	Sales
57	King	Sales
58	Sully	Sales
59	McEwen	Sales
60	Smith	Sales

61	Doran	Sales
62	Sewall	Sales
63	Vishney	Sales
64	Greene	Sales
65	Marvins	Sales
66	Lee	Sales
67	Ande	Sales
68	Banda	Sales
69	Ozer	Sales
70	Bloom	Sales
71	Fox	Sales
72	Smith	Sales
73	Bates	Sales
74	Kumar	Sales
75	Abel	Sales
76	Hutton	Sales
77	Taylor	Sales
78	Livingston	Sales
79	Johnson	Sales
80	Taylor	Shipping
81	Fleur	Shipping
82	Sullivan	Shipping
83	Geoni	Shipping
84	Sarchand	Shipping
85	Bull	Shipping
86	Dellinger	Shipping
87	Cabrio	Shipping
88	Chung	Shipping
89	Dilly	Shipping
90	Gates	Shipping
91	Perkins	Shipping
92	Bell	Shipping
93	Everett	Shipping
94	McCain	Shipping
95	Jones	Shipping
96	Walsh	Shipping
97	Feeney	Shipping
98	OConnell	Shipping
99	Grant	Shipping
100	Whalen	Administration

101	Hartstein	Marketing
102	Fay	Marketing
103	Mavris	Human Resourc...
104	Baer	Public Relations
105	Higgins	Accounting
106	Gietz	Accounting

- 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';

```

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

## Additional exercises

- 1A. 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;

```

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

- 2A. 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	Uman	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

	first_name	last_name	hire_date
21	James	Marlow	1997-02-16 00:00:00.000
22	TJ	Olson	1999-04-10 00:00:00.000
23	Michael	Rogers	1998-08-26 00:00:00.000
24	Ki	Gee	1999-12-12 00:00:00.000
25	Hazel	Philtanker	2000-02-06 00:00:00.000
26	Stephen	Stiles	1997-10-26 00:00:00.000
27	John	Seo	1998-02-12 00:00:00.000
28	Joshua	Patel	1998-04-06 00:00:00.000
29	Randall	Matos	1998-03-15 00:00:00.000
30	Peter	Vargas	1998-07-09 00:00:00.000
31	Alberto	Errazuriz	1997-03-10 00:00:00.000
32	Gerald	Cambrault	1999-10-15 00:00:00.000
33	Eleni	Zlotkey	2000-01-29 00:00:00.000
34	Peter	Tucker	1997-01-30 00:00:00.000
35	David	Bernstein	1997-03-24 00:00:00.000
36	Peter	Hall	1997-08-20 00:00:00.000
37	Christopher	Olsen	1998-03-30 00:00:00.000
38	Nanette	Cambrault	1998-12-09 00:00:00.000
39	Oliver	Tuvault	1999-11-23 00:00:00.000
40	Lindsey	Smith	1997-03-10 00:00:00.000

	first_name	last_name	hire_date
41	Louise	Doran	1997-12-15 00:00:00.000
42	Sarath	Sewall	1998-11-03 00:00:00.000
43	Clara	Vishney	1997-11-11 00:00:00.000
44	Danielle	Greene	1999-03-19 00:00:00.000
45	Mattea	Marvins	2000-01-24 00:00:00.000
46	David	Lee	2000-02-23 00:00:00.000
47	Sundar	Ande	2000-03-24 00:00:00.000
48	Amit	Banda	2000-04-21 00:00:00.000
49	Lisa	Ozer	1997-03-11 00:00:00.000
50	Hamison	Bloom	1998-03-23 00:00:00.000
51	Taylor	Fox	1998-01-24 00:00:00.000
52	William	Smith	1999-02-23 00:00:00.000
53	Elizabeth	Bates	1999-03-24 00:00:00.000
54	Sundita	Kumar	2000-04-21 00:00:00.000
55	Alyssa	Hutton	1997-03-19 00:00:00.000
56	Jonathon	Taylor	1998-03-24 00:00:00.000
57	Jack	Livingston	1998-04-23 00:00:00.000
58	Kimberely	Grant	1999-05-24 00:00:00.000
59	Charles	Johnson	2000-01-04 00:00:00.000
60	Winston	Taylor	1998-01-24 00:00:00.000

61	Jean	Fleur	1998-02-23 00:00:00.000
62	Martha	Sullivan	1999-06-21 00:00:00.000
63	Girard	Geoni	2000-02-03 00:00:00.000
64	Alexis	Bull	1997-02-20 00:00:00.000
65	Julia	Dellinger	1998-06-24 00:00:00.000
66	Anthony	Cabrio	1999-02-07 00:00:00.000
67	Kelly	Chung	1997-06-14 00:00:00.000
68	Jennifer	Dilly	1997-08-13 00:00:00.000
69	Timothy	Gates	1998-07-11 00:00:00.000
70	Randall	Perkins	1999-12-19 00:00:00.000
71	Britney	Everett	1997-03-03 00:00:00.000
72	Samuel	McCain	1998-07-01 00:00:00.000
73	Vance	Jones	1999-03-17 00:00:00.000
74	Alana	Walsh	1998-04-24 00:00:00.000
75	Kevin	Feeney	1998-05-23 00:00:00.000
76	Donald	OConnell	1999-06-21 00:00:00.000
77	Douglas	Grant	2000-01-13 00:00:00.000
78	Pat	Fay	1997-08-17 00:00:00.000

3A. 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	Frapp	1997-04-10 00:00:00.000
5	Alexis	Bull	1997-02-20 00:00:00.000	Adam	Frapp	1997-04-10 00:00:00.000
6	James	Marlow	1997-02-16 00:00:00.000	Adam	Frapp	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	Cambraut	1999-10-15 00:00:00.000
21	Harrison	Bloom	1998-03-23 00:00:00.000	Gerald	Cambraut	1999-10-15 00:00:00.000
22	Taylor	Fox	1998-01-24 00:00:00.000	Gerald	Cambraut	1999-10-15 00:00:00.000
23	William	Smith	1999-02-23 00:00:00.000	Gerald	Cambraut	1999-10-15 00:00:00.000
24	Elizabeth	Bates	1999-03-24 00:00:00.000	Gerald	Cambraut	1999-10-15 00:00:00.000
25	Ellen	Abel	1996-05-11 00:00:00.000	Eleni	Zlotkey	2000-01-29 00:00:00.000
26	Alyssa	Hutton	1997-03-19 00:00:00.000	Eleni	Zlotkey	2000-01-29 00:00:00.000
27	Jonathon	Taylor	1998-03-24 00:00:00.000	Eleni	Zlotkey	2000-01-29 00:00:00.000
28	Jack	Livingston	1998-04-23 00:00:00.000	Eleni	Zlotkey	2000-01-29 00:00:00.000
29	Kimberely	Grant	1999-05-24 00:00:00.000	Eleni	Zlotkey	2000-01-29 00:00:00.000
30	Charles	Johnson	2000-01-04 00:00:00.000	Eleni	Zlotkey	2000-01-29 00:00:00.000

4A. 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	21	206	Gietz	8300.00
2	203	Mavris	6500.00	22	177	Livingston	8400.00
3	165	Lee	6800.00	23	176	Taylor	8600.00
4	113	Popp	6900.00	24	175	Hutton	8800.00
5	155	Tuvault	7000.00	25	109	Faviet	9000.00
6	161	Sewall	7000.00	26	103	Hunold	9000.00
7	178	Grant	7000.00	27	152	Hall	9000.00
8	164	Marvins	7200.00	28	158	McEwen	9000.00
9	172	Bates	7300.00	29	157	Sully	9500.00
10	171	Smith	7400.00	30	151	Bernstein	9500.00
11	160	Doran	7500.00	31	163	Greene	9500.00
12	154	Cambrault	7500.00	32	170	Fox	9600.00
13	111	Sciarra	7700.00	33	169	Bloom	10000.00
14	112	Uman	7800.00	34	204	Baer	10000.00
15	122	Kaufling	7900.00	35	150	Tucker	10000.00
16	120	Weiss	8000.00	36	156	King	10000.00
17	159	Smith	8000.00	37	162	Vishney	10500.00
18	153	Olsen	8000.00	38	149	Zlotkey	10500.00
19	121	Fripp	8200.00	39	148	Cambrault	11000.00
20	110	Chen	8200.00	40	114	Raphaely	11000.00

41	174	Abel	11000.00
42	168	Ozer	11500.00
43	205	Higgins	12000.00
44	108	Greenb...	12000.00
45	147	Errazuriz	12000.00
46	201	Hartstein	13000.00
47	146	Partners	13500.00
48	145	Russell	14000.00
49	101	Kochhar	17000.00
50	102	De Haan	17000.00
51	100	King	24000.00

5A. 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	Uman
6	113	Popp

6A. 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	Erazuriz	12000.00
12	Cambrault	11000.00
13	Zlotkey	10500.00
14	Hartstein	13000.00

7A. 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;
```



	New salary		
1	The salary of King after a 10% raise is 26400.00000	21	The salary of Weiss after a 10% raise is 8800.00000
2	The salary of Kochhar after a 10% raise is 18700.00000	22	The salary of Fripp after a 10% raise is 9020.00000
3	The salary of De Haan after a 10% raise is 18700.00000	23	The salary of Kaufling after a 10% raise is 8690.00000
4	The salary of Hunold after a 10% raise is 9900.00000	24	The salary of Vollman after a 10% raise is 7150.00000
5	The salary of Ernst after a 10% raise is 6600.00000	25	The salary of Mourgos after a 10% raise is 6380.00000
6	The salary of Austin after a 10% raise is 5280.00000	26	The salary of Nayer after a 10% raise is 3520.00000
7	The salary of Pataballa after a 10% raise is 5280.00000	27	The salary of Mikkilineni after a 10% raise is 2970.00000
8	The salary of Lorentz after a 10% raise is 4620.00000	28	The salary of Landry after a 10% raise is 2640.00000
9	The salary of Greenberg after a 10% raise is 13200.00000	29	The salary of Markle after a 10% raise is 2420.00000
10	The salary of Faviet after a 10% raise is 9900.00000	30	The salary of Bissot after a 10% raise is 3630.00000
11	The salary of Chen after a 10% raise is 9020.00000	31	The salary of Atkinson after a 10% raise is 3080.00000
12	The salary of Sciarra after a 10% raise is 8470.00000	32	The salary of Marlow after a 10% raise is 2750.00000
13	The salary of Uman after a 10% raise is 8580.00000	33	The salary of Olson after a 10% raise is 2310.00000
14	The salary of Popp after a 10% raise is 7590.00000	34	The salary of Mallin after a 10% raise is 3630.00000
15	The salary of Raphaely after a 10% raise is 12100.00000	35	The salary of Rogers after a 10% raise is 3190.00000
16	The salary of Khoo after a 10% raise is 3410.00000	36	The salary of Gee after a 10% raise is 2640.00000
17	The salary of Baida after a 10% raise is 3190.00000	37	The salary of Philtanker after a 10% raise is 2420.00000
18	The salary of Tobias after a 10% raise is 3080.00000	38	The salary of Ladwig after a 10% raise is 3960.00000
19	The salary of Himuro after a 10% raise is 2860.00000	39	The salary of Stiles after a 10% raise is 3520.00000
20	The salary of Colmenares after a 10% raise is 2750.00000	40	The salary of Seo after a 10% raise is 2970.00000
41	The salary of Patel after a 10% raise is 2750.00000		
42	The salary of Rajs after a 10% raise is 3850.00000		
43	The salary of Davies after a 10% raise is 3410.00000		
44	The salary of Matos after a 10% raise is 2860.00000		
45	The salary of Vargas after a 10% raise is 2750.00000		
46	The salary of Taylor after a 10% raise is 3520.00000		
47	The salary of Fleaur after a 10% raise is 3410.00000		
48	The salary of Sullivan after a 10% raise is 2750.00000		
49	The salary of Geoni after a 10% raise is 3080.00000	51	The salary of Jones after a 10% raise is 3080.00000
50	The salary of Sarchand after a 10% raise is 4620.00000	52	The salary of Walsh after a 10% raise is 3410.00000
51	The salary of Bull after a 10% raise is 4510.00000	53	The salary of Feeney after a 10% raise is 3300.00000
52	The salary of Dellinger after a 10% raise is 3740.00000	54	The salary of OConnell after a 10% raise is 2860.00000
53	The salary of Cabrio after a 10% raise is 3300.00000	55	The salary of Grant after a 10% raise is 2860.00000
54	The salary of Chung after a 10% raise is 4180.00000	56	The salary of Whalen after a 10% raise is 4840.00000
55	The salary of Dilly after a 10% raise is 3960.00000	57	The salary of Hartstein after a 10% raise is 14300.00000
56	The salary of Gates after a 10% raise is 3190.00000	58	The salary of Fay after a 10% raise is 6600.00000
57	The salary of Perkins after a 10% raise is 2750.00000	59	The salary of Mavris after a 10% raise is 7150.00000
58	The salary of Bell after a 10% raise is 4400.00000	60	The salary of Baer after a 10% raise is 11000.00000
59	The salary of Everett after a 10% raise is 4290.00000	71	The salary of Higgins after a 10% raise is 13200.00000
60	The salary of McCain after a 10% raise is 3520.00000	72	The salary of Gietz after a 10% raise is 9130.00000

## 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

```

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.

```

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	Emst	IT_PROG	6000.00
6	IT	1400	Austin	IT_PROG	4800.00
7	IT	1400	Pataballa	IT_PROG	4800.00
8	IT	1400	Lorentz	IT_PROG	4200.00
9	Finance	1700	Greenberg	FI_MGR	12000.00
10	Finance	1700	Faviet	FI_ACCOUNT	9000.00
11	Finance	1700	Chen	FI_ACCOUNT	8200.00
12	Finance	1700	Sciarra	FI_ACCOUNT	7700.00
13	Finance	1700	Uman	FI_ACCOUNT	7800.00
14	Finance	1700	Popp	FI_ACCOUNT	6900.00
15	Purchasing	1700	Raphaely	PU_MAN	11000.00
16	Purchasing	1700	Khoo	PU_CLERK	3100.00
17	Purchasing	1700	Baida	PU_CLERK	2900.00
18	Purchasing	1700	Tobias	PU_CLERK	2800.00
19	Purchasing	1700	Himuro	PU_CLERK	2600.00
20	Purchasing	1700	Colmenares	PU_CLERK	2500.00
21	Shipping	1500	Weiss	ST_MAN	8000.00

21	Shipping	1500	Weiss	ST_MAN	8000.00
22	Shipping	1500	Fripp	ST_MAN	8200.00
23	Shipping	1500	Kauffling	ST_MAN	7900.00
24	Shipping	1500	Vollman	ST_MAN	6500.00
25	Shipping	1500	Mourgos	ST_MAN	5800.00
26	Shipping	1500	Nayer	ST_CLERK	3200.00
27	Shipping	1500	Mikkilineni	ST_CLERK	2700.00
28	Shipping	1500	Landry	ST_CLERK	2400.00
29	Shipping	1500	Markle	ST_CLERK	2200.00
30	Shipping	1500	Bissot	ST_CLERK	3300.00
31	Shipping	1500	Atkinson	ST_CLERK	2800.00
32	Shipping	1500	Marlow	ST_CLERK	2500.00
33	Shipping	1500	Olson	ST_CLERK	2100.00
34	Shipping	1500	Mallin	ST_CLERK	3300.00
35	Shipping	1500	Rogers	ST_CLERK	2900.00
36	Shipping	1500	Gee	ST_CLERK	2400.00
37	Shipping	1500	Philtanker	ST_CLERK	2200.00
38	Shipping	1500	Ladwig	ST_CLERK	3600.00
39	Shipping	1500	Stiles	ST_CLERK	3200.00
40	Shipping	1500	Seo	ST_CLERK	2700.00

41	Shipping	1500	Patel	ST_CLERK	2500.00
42	Shipping	1500	Rajs	ST_CLERK	3500.00
43	Shipping	1500	Davies	ST_CLERK	3100.00
44	Shipping	1500	Matos	ST_CLERK	2600.00
45	Shipping	1500	Vargas	ST_CLERK	2500.00
46	Sales	2500	Russell	SA_MAN	14000.00
47	Sales	2500	Partners	SA_MAN	13500.00
48	Sales	2500	Errazuriz	SA_MAN	12000.00
49	Sales	2500	Cambrault	SA_MAN	11000.00
50	Sales	2500	Zlotkey	SA_MAN	10500.00
51	Sales	2500	Tucker	SA_REP	10000.00
52	Sales	2500	Bemstein	SA_REP	9500.00
53	Sales	2500	Hall	SA_REP	9000.00
54	Sales	2500	Olsen	SA_REP	8000.00
55	Sales	2500	Cambrault	SA_REP	7500.00
56	Sales	2500	Tuvault	SA_REP	7000.00
57	Sales	2500	King	SA_REP	10000.00
58	Sales	2500	Sully	SA_REP	9500.00
59	Sales	2500	McEwen	SA_REP	9000.00
60	Sales	2500	Smith	SA_REP	8000.00
61	Sales	2500	Doran	SA_REP	7500.00
62	Sales	2500	Sewall	SA_REP	7000.00
63	Sales	2500	Vishney	SA_REP	10500.00
64	Sales	2500	Greene	SA_REP	9500.00
65	Sales	2500	Marvins	SA_REP	7200.00
66	Sales	2500	Lee	SA_REP	6800.00
67	Sales	2500	Ande	SA_REP	6400.00
68	Sales	2500	Banda	SA_REP	6200.00
69	Sales	2500	Ozer	SA_REP	11500.00
70	Sales	2500	Bloom	SA_REP	10000.00
71	Sales	2500	Fox	SA_REP	9600.00
72	Sales	2500	Smith	SA_REP	7400.00
73	Sales	2500	Bates	SA_REP	7300.00
74	Sales	2500	Kumar	SA_REP	6100.00
75	Sales	2500	Abel	SA_REP	11000.00
76	Sales	2500	Hutton	SA_REP	8800.00
77	Sales	2500	Taylor	SA_REP	8600.00
78	Sales	2500	Livingston	SA_REP	8400.00
79	Sales	2500	Johnson	SA_REP	6200.00
80	Shipping	1500	Taylor	SH_CLERK	3200.00

81	Shipping	1500	Fleaur	SH_CLERK	3100.00
82	Shipping	1500	Sullivan	SH_CLERK	2500.00
83	Shipping	1500	Geoni	SH_CLERK	2800.00
84	Shipping	1500	Sarchand	SH_CLERK	4200.00
85	Shipping	1500	Bull	SH_CLERK	4100.00
86	Shipping	1500	Dellinger	SH_CLERK	3400.00
87	Shipping	1500	Cabrio	SH_CLERK	3000.00
88	Shipping	1500	Chung	SH_CLERK	3800.00
89	Shipping	1500	Dilly	SH_CLERK	3600.00
90	Shipping	1500	Gates	SH_CLERK	2900.00
91	Shipping	1500	Perkins	SH_CLERK	2500.00
92	Shipping	1500	Bell	SH_CLERK	4000.00
93	Shipping	1500	Everett	SH_CLERK	3900.00
94	Shipping	1500	McCain	SH_CLERK	3200.00
95	Shipping	1500	Jones	SH_CLERK	2800.00
96	Shipping	1500	Walsh	SH_CLERK	3100.00
97	Shipping	1500	Feeney	SH_CLERK	3000.00
98	Shipping	1500	OConnell	SH_CLERK	2600.00
99	Shipping	1500	Grant	SH_CLERK	2600.00
100	Administration	1700	Whalen	AD_ASST	4400.00
100	Administration	1700	Whalen	AD_ASST	4400.00
101	Marketing	1800	Hartstein	MK_MAN	13000.00
102	Marketing	1800	Fay	MK_REP	6000.00
103	Human Resourc...	2400	Mavris	HR_REP	6500.00
104	Public Relations	2700	Baer	PR_REP	10000.00
105	Accounting	1700	Higgins	AC_MGR	12000.00
106	Accounting	1700	Gietz	AC_ACCOU...	8300.00

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
```

	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
```

	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);
```

	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
);
```

	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
);
```

	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;
```

	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
```



Results			Mess		
	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	28	2014	98
23	2009	98	29	2015	98
24	2010	98	30	2016	98
25	2011	98	31	2017	98
26	2012	98	32	2018	98
27	2013	98	33	2019	98

10. 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
```

	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.

```

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
);

```

	last_name						
1	Emst	21	Marlow	41	King	61	Taylor
2	Austin	22	Olson	42	Sully	62	Livingston
3	Pataballa	23	Mallin	43	McEwen	63	Grant
4	Lorentz	24	Rogers	44	Smith	64	Johnson
5	Faviet	25	Gee	45	Doran	65	Taylor
6	Chen	26	Philtanker	46	Sewall	66	Fleur
7	Sciarra	27	Ladwig	47	Vishney	67	Sullivan
8	Uman	28	Stiles	48	Greene	68	Geoni
9	Popp	29	Seo	49	Marvins	69	Sarchand
10	Khoo	30	Patel	50	Lee	70	Bull
11	Baida	31	Rajs	51	Ande	71	Dellinger
12	Tobias	32	Davies	52	Banda	72	Cabrio
13	Himuro	33	Matos	53	Ozer	73	Chung
14	Colmenares	34	Vargas	54	Bloom	74	Dilly
15	Nayer	35	Tucker	55	Fox	75	Gates
16	Mikkilineni	36	Bernstein	56	Smith	76	Perkins
17	Landry	37	Hall	57	Bates	77	Bell
18	Markle	38	Olsen	58	Kumar	78	Everett
19	Bissot	39	Cambrault	59	Abel	79	McCain
20	Atkinson	40	Tuvault	60	Hutton	80	Jones
						81	Walsh
						82	Feeney
						83	OConnell
						84	Grant
						85	Whalen
						86	Fay
						87	Mavris
						88	Baer
						89	Gietz

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.

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	21	Nayer	41	Lorentz		
2	Khoo	22	Mikkilineni	42	Olsen		
3	Baida	23	Landry	43	Cambrault		
4	Tobias	24	Markle	44	Tuvault		
5	Himuro	25	Bissot	45	Smith		
6	Colmenares	26	Atkinson	46	Doran		
7	Taylor	27	Marlow	47	Sewall		
8	Fleaur	28	Olson	48	Marvins		
9	Sullivan	29	Mallin	49	Lee		
10	Geoni	30	Rogers	50	Ande		
11	Dellinger	31	Gee	51	Banda		
12	Cabrio	32	Philtanker	52	Smith		
13	Gates	33	Stiles	53	Bates		
14	Perkins	34	Seo	54	Kumar		
15	McCain	35	Patel	55	Hutton		
16	Jones	36	Davies	56	Taylor	61	Chen
17	Walsh	37	Matos	57	Livingston	62	Sciarra
18	Feeney	38	Vargas	58	Johnson	63	Uman
19	OConnell	39	Austin	59	Kochhar	64	Popp
20	Grant	40	Pataballa	60	De Haan	65	Gietz

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;

```

	last_name				
1	Faviet	21	Davies	41	Smith
2	Sciarra	22	Matos	42	Bates
3	Tobias	23	Vargas	43	Abel
4	Weiss	24	Tucker	44	Hutton
5	Kaufling	25	Bernstein	45	Taylor
6	Nayer	26	Hall	46	Livingston
7	Mikkilineni	27	Olsen	47	Johnson
8	Landry	28	Cambrault	48	Taylor
9	Bissot	29	Tuvault	49	Fleaur
10	Atkinson	30	King	50	Sullivan
11	Marlow	31	Sully	51	Sarchand
12	Olson	32	McEwen	52	Bull
13	Mallin	33	Smith	53	Dellinger
14	Rogers	34	Doran	54	Cabrio
15	Gee	35	Sewall	55	Chung
16	Ladwig	36	Vishney	56	Dilly
17	Stiles	37	Greene	57	Gates
18	Seo	38	Marvins	58	Perkins
19	Patel	39	Bloom	59	Bell
20	Rajs	40	Fox	60	Everett
				61	McCain
				62	Jones
				63	Walsh
				64	Feeney
				65	OConnell
				66	Grant

7A. Display the department names of those departments whose total salary cost is above one-eighth (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
)
```

	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