oracle2

작성일시	@2022년 8월 8일 오후 5:50
강의 번호	
유형	
자료	
복습	☐

ORACLE

그룹함수 - 단일행 함수와는 달리 여러 행에 대해 함수가 적용되어 하나의 결과를 나타내는 함수이다. 예를 들면 학생들의 영어점수들의 합계, 수학점수들의 평균…. 열에 대해 같은 값끼리 묶어서 묶은 행들의 집합에 대해 그룹 연산이 필요할때는 GROUP BY 절을 사용하여 처리하고 또 묶은 그룹에 다시 조건이 필요할때는 HAVING절을 이용한다.

SELECT 그룹함수(열이름)

FROM 테이블이름

[HVING 조건식]

[ORDER BY 열이름];

COUNT 개수 null의 값도 개수로 셈

SUM 합계 나머지 아래 함수들은 null값은 제외하고 연산을 수행한다.

AVG 평균

MAX 최대값

MIN 최소값

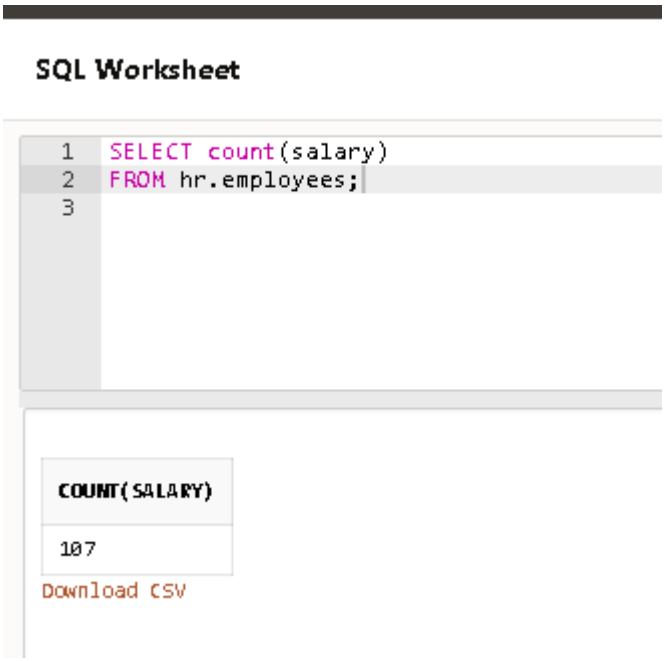
SEDDEV 표준편자

VARIANCE 분산

Quiz 1

employees 테이블에서 salary 의 행의 개수가 몇개인지 세어서 출력하세요

- 주의 -count함수는 null값도 행으로 셀수 있다는 점을 유의하자

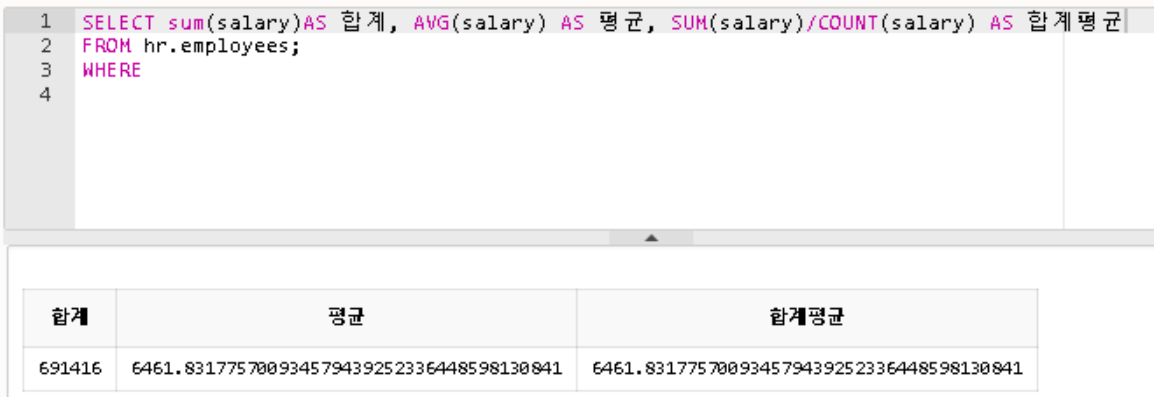


Quiz 2

employees 테이블에서 salary 의 합계와 평균을 구하세요 또한 avg함수를 사용하지말고

salary 의 평균을 구하세요

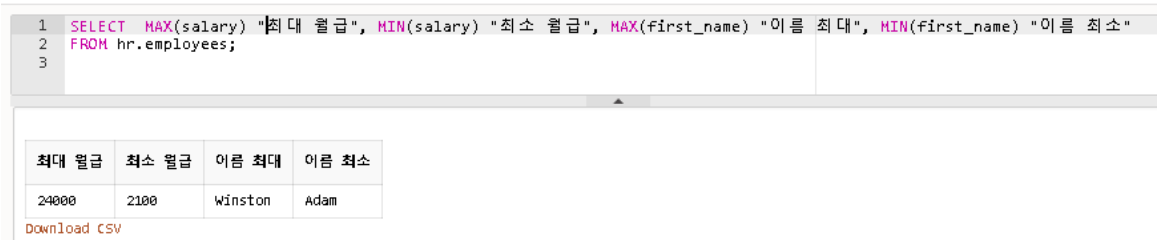
- AVG 함수는 null값을 제외하고 평균을 구하는데 null 값을 포함하여 평균을 계산해야 한다면 뒤에서 배울 NVL 함수를 사용하여 구할 수있다.



Quiz 3

employees 테이블에서 salary의 최대값과 최소값 first_name의 최대값과 최소값을 출력하세요

최대 월급 최소 월급 이름 최대 이름 최소



```
SELECT  MAX(salary) "최대 월급", MIN(salary) "최소 월급", MAX(first_name) "이름 최대", MIN(first_name) "이름 최소"
FROM hr.employees;
```

최대 월급	최소 월급	이름 최대	이름 최소
24000	2100	Winston	Adam

Download CSV

- GROUP BY

  특정 열의 데이터 값들을 그룹으로 묶어 그룹함수를 적용한다.

  SELECT 절에 열이름과 그룹 함수를 함께 기술할때는 GROUP BY 절을 사용하게 된다. 그룹화는 열이름이 기술된 순서대로 수행된다.

  SELECT 기준열, 그룹함수(열이름)

  FROM 테이블이름

  GROUP BY 열이름

  테이블에 접근하여

  기술된 열이름을 기준으로 그룹화하여

  선택 얄들을 나열하고 지정된 열이름에 그룹홤수를 적용하게 된다.

  Quiz 4

  employees 테이블에서 employee_id 가 10이상인 직운들에 대해서 job_id 별로 그룹화하여 job_id 별 총 급여와 job_id별 평균 급여를 구하고 job_id 별 총 급여를 기준으로 내림차순 정렬하세요.

  직무 직무별 총급여 직무별 평균 급여

```sql
SELECT job_id, sum(salary) AS "직무별 총 급여 ", AVG(salary) "직무별 평균급여"
FROM hr.employees
WHERE employee_id >=10
GROUP BY job_id
ORDER BY job_id DESC;
```

JOB_ID	직무별 총 급여	직무별 평균급여
ST_MAN	36400	7280
ST_CLERK	55700	2785
SH_CLERK	64300	3215
SA_REP	250500	8350
SA_MAN	61000	12200
PU_MAN	11000	11000
PU_CLERK	13900	2780
PR_REP	10000	10000

- HAVING

  그룹화된 값에 조건식을 적용할때 사용된다. 즉WHERE절에서 그룹함수를 사용할수 없기때문에 HAVING 절을 사용하여 그룹함수의 결과값에 대해 조건식을 적용할수 있다.

  SELECT 기준열, 그룹함수(열이름)

  FROM 테이블이름

  [WHERE 조건식]

  GROUP BY 열이름

  [HAVING 조건식] ;

Quiz 5

employees 테이블에서 employee_id가 10이상인 직원에 대해서 job_id 별로 그룹화하여 job_id 별 총급여와 job_id별 평균 급여를 구하되

job_id 별로 총급여가 300000보다 큰 값 만 출력하는데 결과는 job_id별 총 급여를 기준으로 내림차순 정렬하세요

직무 직무별 총급여 직무별 평균급여

```
1  SELECT job_id AS 직무, sum(salary) AS "직무별 총 급여 ", AVG(salary) "직무별 평균급여"
2  FROM hr.employees
3  WHERE employee_id >=10
4  GROUP BY job_id
5  HAVING sum(salary) > 30000
6  ORDER BY sum(salary) DESC;
7
```

직무	직무별 총 급여	직무별 평균급여
SA_REP	250500	8350
SH_CLERK	64300	3215
SA_MAN	61000	12200
ST_CLERK	55700	2785
FI_ACCOUNT	39600	7920
ST_MAN	36400	7280
AD_VP	34000	17000

조인JOIN

관계형 데이터베이스의 의미는 데이블들이 관계를 맺고 조작되는 원리에서 유래 되었다

테이블들에는 각각의 성질에 맞는 데이터들이 저장되 있고 그 테이블들은 틀정한 규칙에 따라 상호 관계를 맺는다. 데이터는 여러 테이블에 흩어져 저장되어 있어 사용자가 원하는 대로 데이터를 사용하려면 특별한 방법이 필요하다. 이 때 사용되는 기법이 바로 조인이다.

조인은 한개 이상의 테이블과 테이블을 서로 연결하여 사용하는 기법을 의미한다.

실무에서는 동등 조인, 외부 조인, 자체 조인, 등을 사용하게 된다.

(곱집합, 비동등조인… 다양한 조인들도 있다.)

SELECT 테이블명1, 열이름, 테이블명2, 열이름2

FROM 테이블명1, 테이블명2

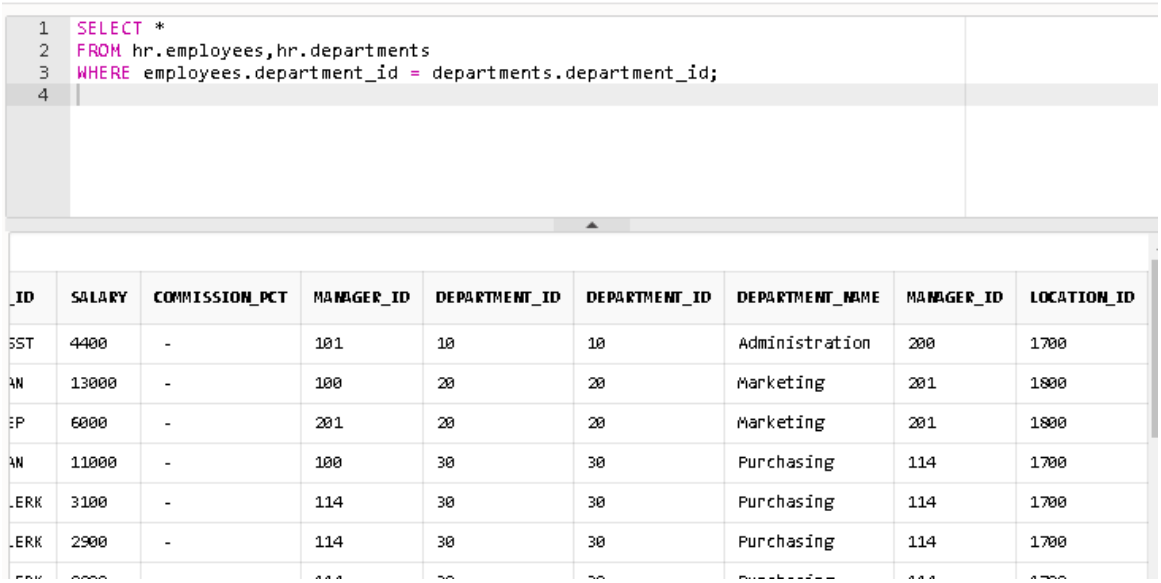
WHERE 테이블명1,열이름1 = 테이블명2,열이름2;

조인을 사용할때 규칙

- SELECT 절에는 출력할 열이름들을 모두 기술한다.
- FROM 절에는 접근하려는 테이블들을 기술한다.
- WHERE 절에는 조인 조건을 기술한다.
- 사용되는 테이블이름들에는 별칭을 alias 을 사용하면 편하다.

1. 동등 조인 - 똑같은 데이터 끼리 연결

   동등 조인은 양 테이블에서 조인 조건이일치하는 행들만 가져오는 가장 일반적으로 자주 사용되는 조인이다. = 등호 연산자를 사용하여 조건 값이 정확하게 일치하때만 행을 가져 오기 때문에 inner join 이라고도 부른다.



```
1  SELECT *
2  FROM hr.employees,hr.departments
3  WHERE employees.department_id = departments.department_id;
4
```

_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
SST	4400	-	101	10	10	Administration	200	1700
AN	13000	-	100	20	20	Marketing	201	1800
EP	6000	-	201	20	20	Marketing	201	1800
AN	11000	-	100	30	30	Purchasing	114	1700
LERK	3100	-	114	30	30	Purchasing	114	1700
LERK	2900	-	114	30	30	Purchasing	114	1700

```
1  SELECT *
2  FROM hr.departments;
3
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400

```
1  SELECT *
2  FROM hr.employees;
```

ST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
g	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
har	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
aan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
old	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60

Quiz

employees 테이블과 dapartments 테이블과 locations 테이블을 조인하여 각 지원이 어느 부서에 속하는 지와 부서의 소재지가 어디인지 조회하시오

```
SELECT employees.employee_id 사원아이디, departments.department_id 부서아이디, departments.department_name 부서명,
departments.location_id 지역아이디, locations.city 도시
FROM hr.employees, hr.departments, hr.locations
WHERE employees.department_id = departments.department_id
AND departments.location_id=locations.location_id;
```

사원아이디	부서아이디	부서명	지역아이디	도시
100	90	Executive	1700	Seattle
101	90	Executive	1700	Seattle
102	90	Executive	1700	Seattle
103	60	IT	1400	Southlake

```
SELECT *
FROM hr.employees;
```

ATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
-03	AD_PRES	24000	-	-	90
-05	AD_VP	17000	-	100	90
-01	AD_VP	17000	-	100	90
-06	IT_PROG	9000	-	102	60

```
SELECT *
FROM hr.departm
```

```
SELECT *
FROM hr.locations;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	24

LOCATION_ID	STREET_ADDRESS
1000	1297 Via Cola di ie
1100	93091 Calle
1200	2017 Shinjuku-ku
1300	9450 Kamiya-cho
1400	2014 Jabberwocky d

외부 조인 - 모든 데이터를 연결

동등조인은 데이터 값이 정확히 일치하는 경우에만 결과를 출력한다. 데이터 값이 일치하지 않으면 결과가 조회되지 않는다.

```
1  SELECT count(*) "조인된 건수"
2  FROM hr.employees E, hr.departments D
3  WHERE E.department_id = D.department_id;
```

조인된 건수
106

```
1  SELECT count(*) "건수"
2  FROM hr.employees E;
```

건수
107

동등 조인의 결과는 106건이 출력되었는데 employees 의 직원정보는 모두 107이다 1건의 차이가 나는 이유는 일치하지 않는 1건이 누락되었기 때문이다.

외부 조인은(outer join) 조건을 만족하지 않는 행도 모두 출력하는 조인 기법이다.

- 외부 조인은 동등 조인 조건을 만족하지 못해 누락되는 행을 출력하기 위해 사용된다.
- 외부 조인은(+) 기호를 사용한다. +기호는 조인할 행이 없는 즉 데이터가 부족한 데이블의 열 이름 뒤에 기술한다
- + 기호는 외부 조인 하려는 한쪽에만 기술할 수 있다 테이블 양쪽에 모두 기술할수는 없다.

- + 기호를 붙이면 데이터 값이 부족한 테이블에 null 값을 갖는 행이 생성되어 테이더 값이 충분한 테이블의 행들이 null행에 조인된다.

```
1  SELECT count(*) "조인된 건수"
2  FROM hr.employees E, hr.departments D
3  WHERE E.department_id = D.department_id(+);
```

조인된 건수
107

외부 조인은 동등 조인과 함께 실무에서 가장 많이 사용되는 조인이다.

- 양쪽 테이블 중 전부 출력하고 싶은 테이블 쪽을 앞에둔다
- (+)는 다른쪽(뒤쪽) 테이블에 조인 조건에 붙인다.

3. 자체 조인 self join

   employees 테이블의 직원 정보에는 manager_id 열이 있다. 이 열은 그 직원의 담당 매니저의 정보를 담고 있는 열이다. 그러면 생각해보자. 어떤 직원의 담당 매니저를 알고싶으면 그 테이블에 결국 다시 조인해야한다. 이렇게 자기 자신의 테이블을 조인하는 것을 자체 조인 self join 이라고 한다.

   SELECT 열이름1,열이름2, 열이름3

   FROM 테이블1 별명1, 테이블1 별명2

   WHERE 테이블.열이름1 = 테이블.열이름2

   한개의 테이블의 별명을 2개로 각각 다른 이름으로 설정한다.

Quiz

employees 테이블을 자체조인하여 직원별 담당 매니저가 누구인지 조회하세요

사원아이디 이름(first_name + last_name) 매니저 아이디 매니저 이름

```
SELECT a.employee_id 사원아이디, a.first_name || a.last_name 사원이름,
       a.manager_id 매너저아이디, b.first_name || b.last_name 매니저이름
FROM hr.employees a, hr.employees b
WHERE a.manager_id = b.employee_id;
```

사원아이디	사원이름	매너저아이디	매니저이름
101	NeenaKochhar	100	StevenKing
102	LexDe Haan	100	StevenKing
114	DenRaphaely	100	StevenKing
120	MatthewWeiss	100	StevenKing
121	AdamFripp	100	StevenKing
122	PayamKaufling	100	StevenKing
123	ShantaVollman	100	StevenKing
124	KevinMourgos	100	StevenKing

서브 쿼리 - SELECT 문 안에 SELECT

서브쿼리는 SELECT 구문 안에서 보조로 사용되는 또 다른 SELECT 구문이다.

SELECT 구문을 효율족으로 작성할수 있도록 도와준다.

복잡한 SELECT 구문을 작성할 때 필수적으로 사용되는 기법이다.

SELECT 열이름1,열이름2,열이름3.....

FROM 테이블이름

WHERE 조건식

(SELECT 열이름1,열이름2,열이름3.....

FROM 테이블이름

WHERE 조건식);

서브 쿼리는 논리가 복잡한 SLQL 구문에서는 거의 필수로 많이 사용된다

- 서브쿼리 ( ) 괄호 로 묶어서 사용한다. 메인은 묶지 않는다
- 메인쿼리와 서브쿼리를 연결하기위해 단일행 연산자 또는 다중행 연산자를 사용한다.
- 서브쿼리(괄호안)가 먼저 실행되고 그 다음 메인 쿼리가 실행된다,
- 서브쿼리의 서브쿼리의 서브쿼리…… 형태로 계속 중첩시킬수 있다.

서브 쿼리의 종류

- 단일행 서브쿼리 - 하나의 행을 검색하는 서브쿼리
- 다중행 서브쿼리 - 하나 이상의 행을 검색하는 서브쿼리
- 다중열 서브쿼리 - 하나 이상의 열을 검색하는 서브 쿼리

1. 단일행 서브 쿼리 - 서브쿼리 SELECT 문에서 얻은 한개의 행의 결과값을 메인 쿼리로 전달하는 서브쿼리이다.

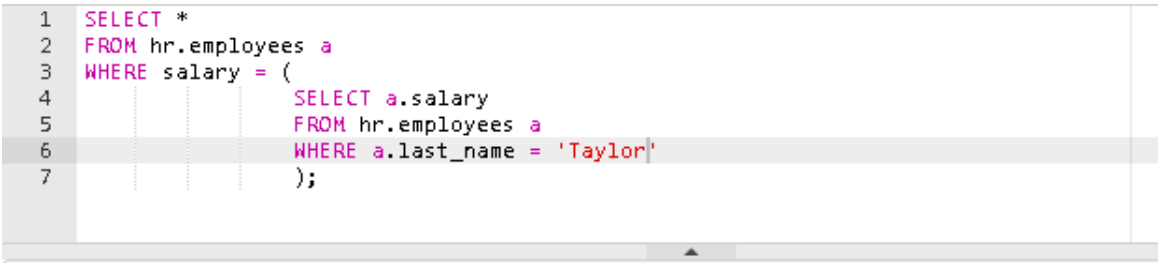
Quiz

employees 테이블의 last_name 이 ‘De Haan’ 인 직원과 salary가 동일한 직원을 단일행 서브쿼리를 이용하여 검색하여 출력하시오

```
SELECT *
FROM hr.employees a
WHERE salary = (
                SELECT a.salary
                FROM hr.employees a
                WHERE a.last_name = 'De Haan'
                );
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000

employees 테이블의 last_name 이 ‘Taylor’ 인 직원과 salary 가 동일한 직원에 대한 모든 정보를 단일행 서브쿼리를 이용하여 검색하여 출력하시오.



```
SELECT *
FROM hr.employees a
WHERE salary = (
                SELECT a.salary
                FROM hr.employees a
                WHERE a.last_name = 'Taylor'
                );
```

Taylor의 결과 값이 2개가 나와 찾지를 못한다.

다중행 서브쿼리

사용법은 단일행과 동일하다 . 다중행 서브쿼리는 하나 이상의 결과를 메인 쿼리에 전달하는 경우에 사용된다.

IN 같은 값이 여러개 IN(1,2,3)

NOT IN 같은 같이 아닌 여러개 NOT IN(1,2,3)

EXIST 값이 있으면 True 없으면 False

ANY 최소한 하나라도 존재 (or) ANY(10,20

ALL 모두 만족 (and) ALL(10,20)

Quiz

employees 테이블에서 department_id 별로 가장 낮은 salary 가 얼마인지 찾아서 그에 해당하는 직원이 누구인지 다중행 서브쿼리를 이용하여 찾아보자.

다중 열 서브쿼리

메인쿼리와 서브 쿼리를 비교하는 WHERE 조건식에서 열이 여러개일 경우 사용되는 서브 쿼리이다.

SELECT 열이름1,열이름2,열이름3

FORM 테이블이름

WHERE (열이름1,열이름2,열이름3….) IN

(SELECT 열이름1,열이름2,열이름3……

FORM 테이블이름

WHERE 조건식);

Quiz

employees 테이블에서 job_id 별로 가장 높은 salary 가 얼마인지 찾아보고 찾아낸 job_id별 salary에 해당하는 직원이누구인지 다중열 서브쿼리를 찾아보자

```sql
SELECT *
FROM hr.employees a
WHERE (a.job_id, a.salary) IN
                (
                SELECT a.job_id, MAX(salary)
                FROM hr.employees a
                GROUP BY a.job_id
                )
ORDER BY a.salary DESC;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000
145	John	Russell	JRUSSEL	011.44.1344.429268	01-OCT-04	SA_MAN	14000
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-04	MK_MAN	13000