Temel SQL Sorguları Üçüncü Kısım

Tarih formatı

Karakterler yanyana olarak ifade edilirse numerik bir çıktı elde edilir (MM,YYYY).

Kelime olarak ifade edilir ise çıktı tarihin kendisi olarak verilir (Month, Day, Year).

Apexde ise format, iki hane ay, iki hane gün ve dört hane de yıl şeklindedir.
(MM/DD/YYYY)

```
SELECT TO_DATE('01-APR-2022', 'DD-MON-YYYY')

FROM DUAL;
```

NOT: Virgülden sonraki kısımda inputun/girdinin formatını belirlemiş olduk.

```
SELECT  TO_CHAR(TO_DATE('01-APR-2022','DD-MON-YYYY'),'DD-MON-YYYY')

FROM DUAL;
```

NOT: Outputun/çıktının formatını düzenledik.

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6),'FRIDAY'),

'fmDay, Month ddth, YYYY') AS "Next Evaluation"

FROM employees;

WHERE employee_id=100;
```

NOT: Şimdiki tarihe 6 ay ekler, bu tarihten itibaren de ilk Cuma gününü bulur. Bu tarihi de karaktere çevirip günü gün olarak verir.

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6),'FRIDAY'),

'DD-MONTH_YYYY')

FROM employees;

WHERE employee_id=100;
```

NULL FUNCTİONS

1) NVL

2) NVL2

3) NULLIF

4) COALESCE ()

1) NVL

Seçtiğimiz kısım null değere sahip ise ona null getirmek yerine vereceğim değeri yazar.

```
SELECT last_name, NVL(commission_pct, 0)
FROM employees
WHERE department_id IN(80,90);
```

```
SELECT NVL(date_of_independence, 'No Date')
FROM wf_countries;
```

```
SELECT last_name, NVL(commission_pct, 0) *250 as "Commission"
FROM employees
WHERE department_id IN(80,90);
```

2) NVL2

Seçtiğim kısımdaki ilk değer null değilse ikinci değeri devreye sokar

NVL: “Verdiğim sütundakı değer nullsa şunu yap.”

NVL2: “Verdiğim sütundakı değer null değilseü çüncü satırı, değilse diğerini yap.”

```
SELECT last_name, salary,
NVL2(commission_pct, salary +(salary*commission_pct),salary) as "İNCOME"
FROM employees
WHERE department_id IN(80,90);
```

NOT: eğer commission_pct null ise yerine salary yazar, değilse hesaplamayı devreye sokar.

3) NULLIF

Length eşitse null yapar, eşit değilse birincinin değerini basar.

```
SELECT first_name, LENGTH(first_name) as length fn, last_name
LENGTH(last_name) as lenght ln, NULLIF(LENGTH(first_name))
FROM employees
WHERE department_id IN(80,90);
```

4) COALESCE

Null olmayan değer bulana kadar kovalar.

```
SELECT last_name, commission_pct, salary
COALESCE(commission_pct, salary, 10)
FROM employees;
```

NOT: commission_pct null değere sahipse salary'yi yazar, o da null değer ise 10 yazar.

NOT: comission_pct null değil direkt yazdı, null olanlarda salary değerini yazdı. 10 yazması için salarynin de null olması gerekirdi.

CONDİTİONAL EXPRESSİONS

DECODE CASE, if else gibi bir karar mekanizmasıdır.

1) **CASE**

CASE expr WHEN comparison_expr1 THEN return_expr1

NOT: expr değişkeni ilkine eşitse ikincsini retrun eder.

SELECT last_name, department_id as "DepNo"

CASE department_id

WHEN 90 THEN 'Management'

WHEN 80 THEN 'Sales'

WHEN 60 THEN 'IT'

ELSE 'Other dept'

END AS "Department"

FROM employees;

2) **DECODE**

Sıralı bir kontrol yapısıdır.

SELECT last_name, department_id as "DepNo"

DECODE (department_id, 90,'Management', 80, 'Sales', 60, 'IT', 'Other dept') AS "Department"

FROM employees;

JOİN

SQL'ın temel aleti JOIN’dir. Dağıttıklarımızı doğru bir şekilde birleştirebilmek için gereklidir.

1) NATURAL JOİN

Kayıpsız bir normalizasyon kontrolü için kullanırız. Aynı adı alan tüm sütunları karşılaştırır. Aynı sütunları bulup içindeki değerleri eşlestirip birleştirir.

a) SELECT first_name, last_name, job_id

FROM employees;

b) SELECT job_id, job_title FROM jobs;

c) SELECT first_name, last_name, job_id, job_title

FROM employees NATURAL JOIN jobs

WHERE department_id>80;

NOTLAR:

// Burada job_ title’ı join ediyorum.

//Bu işlemi sütunları ve içerisindeki değerleri karşılaştırarak uyguluyor.

//desc employees;

//desc jobs;

//ortak sütun job_id

SELECT department_name, city

FROM departments NATURAL JOIN locations;

NOTLAR:

//department ve locations arasında bir join yaptık.

//location ıd uzerınden var olan ortak sütun üzerinden ilişki kuruldu.

//desc departments

//desc locations

2) **CROSS JOIN**

SELECT last_name, department_name

FROM employees CROSS JOIN departments;

SELECT last_name, department_name

FROM employees departments;

NOT: Burada cross joın otomatik yaptı. WHERE veya JOIN ON, JOIN USING, NATURAL JOIN kullanırsam yapmazdı.

JOIN CLAUSES

1) **USING**

Natural Joın aksine burada spesifik olarak sadece belirli bir satırın değerlerini birleştirebiliyoruz. Kısaca sütun seçebiliyoruz.

SELECT first_name, last_name, department_id, department_name

FROM employees JOIN departments USING (department_id);

NOT: 20 row employees da vardı. Bir tanesi null değer olduğu için sonuç 19 çıktı.

NOT: Eğer burada NATURAL JOIN deseydik;

SELECT first_name, last_name, department_id, department_name

FROM employees JOIN departments;

//Yıne 19 urettı. Ortak sutun sadece 1 tane cunku.

//İkisinde de olmayan şeyi usıng dersem hata verir. Usıngde kullancaĞım sütun iki kısımda da ortak olarak olması lazım.

SELECT first_name, last_name, department_id, department_name

FROM employees JOIN departments USING (employee_id);

NOT: USING içinde Alies almaması lazım. Hata verir.

ALİAS ile:

SELECT last_name, e.job_id, job_title

FROM employees e, jobs j

WHERE e.job_id = j.job_id;

NOT: WHERE ile job id'ler eşitse bunu yap dedik ve joın yapmış olduk, Cross joından kurtulduk. Where olmasa 140 satır gelirdi.

SELECT last_name, job_id, job_title

FROM employees e JOIN jobs j USING(job_id)

NOT: e.job_id şeklinde yazarsak hata verir.

NATURAL JOIN VE USING ARASINDAKİ FARK

desc job_history; (employee job ve department ıd si var. Bunlar employee de de avrdı)

SELECT first_name, last_name (sadece employeesda var) , start_date, end_date (historyden gelir)

FROM employees NATURAL JOIN job_history

1.) SELECT first_name, last_name, start_date, end_date

FROM employees JOIN job_history USING (job_id)

2.)SELECT first_name, last_name, start_date, end_date

FROM employees JOIN job_history USING (job_id, employee_id)

3.)SELECT first_name, last_name, start_date, end_date

FROM employees JOIN job_history USING (job_id, employee_id, department_id) (ortakları ekledik)

2) **ON**

Burada sütunun aynı olup olmaması artık önemlı değil. Yani farklı sutünlarda JOIN ON yapabilirim.

APEX ile ilgili;

desc employees; (manager_id var)

desc departments; (yine manager_id vardı eskiden yanlış bir durumdu, joın ornegı gosterebılmek ıcın. Deparmenttaki manager_id, dept_manager_id olarak değiştirildi. İçindeki valuelar eşit)

SELECT employee_id, manager_id, dept_manager_id

FROM employees JOIN departments ON(manager_id=dept_manager_id)

NOT: ON ile sütunu ben söyledim, içindeki değerleri eşleşiyorsa bu sütunu birleştir. Sütun adları farklı ama içindeki değerler aynı.

SELECT employee_id, manager_id, dept_manager_id

FROM employees NATURAL JOIN departments;

NOT: Natural joın ile ne yapardı?

SELECT last_name, job_title

FROM employees e JOIN jobs j ON(e.job_id=j.job_id)

NOT: Birinin job id'si ile diğerininkini eşleştirdi

SELECT last_name, job_title

FROM employees e JOIN jobs j USING(job_id)

NOT: USING ile ortak olanla da getirebildim.

select * from job_grades; (Çalışanların grade'lerini nasıl tespit edeceğim? Aralarında sütun içindeki değerleri aynı olan yok.)

SELECT last_name, salary, grade_level, lowest_sal, highest_sal

FROM employees JOIN job_grades ON(salary=lowest_sal);

NOT: Bu gradeler arasında en düşük kim alıyor?

BİRDEN FAZLA TABLONUN JOIN EDİLMESİ

SELECT last_name department_name as department, city

FROM employees JOIN departments USING(department_id);

USING, NATURAL JOIN ve ON ARASINDAKİ FARK

NATURAL ve USING'de aynı sütun değerleri karşılaştırılırken ON'da ise farklı sütunlar içindeki değerler karşılaştırılıyor.

3) INNER & OUTHER JOIN

INNER JOIN default değerdir. Belirtmeye gerek yok.

SELECT e.last_name, d.ddepartment_id, d.department_name

FROM employees e JOIN departmens d ON (e.department_id=d.department_id);

NOT: 19 kayıt döner çünkü birinin departmanı yok.

a)emp de var dept te olmayanlarda dahil

SELECT e.last_name, d.ddepartment_id, d.department_name

FROM employees e

LEFT OUTHER JOIN departmens d ON (e.department_id=d.department_id);

NOTLAR:

//left outher dedım ve 20 kayıt döndü. Department id si yok ama yine de geldi. Çünkü OUTHER hepsini getirsin, LEFT OUTHER ise eşleşmeyeni de getir demek.

//FROM dan sonra employees gelmiş 1. ve department 2. Left olan employees, left outher ile employeesda olup departmensda olmayanı getır dedım.

a) emp de var dept te olmayanlarda dahil

SELECT e.last_name, d.ddepartment_id, d.department_name

FROM departments e

RIGHT OUTHER JOIN employees d ON (e.department_id=d.department_id);

//üstteki ile aynı sonucu verir.

//Neden Ailes veririz? Department ıd lerı karsılstırablmek ıcın. Department_id ikisindede olduğu için e ve d olarak ayırmam ve onların esıtlıgını boyle kontrol edebılmem lazım

//Left dedıgı zaman employeesa baktı, employeesda olanlar deparetmanta olmasa bile getir dedim.

b) dept de var empt de olmayanlar da dahil

SELECT e.last_name, d.ddepartment_id, d.department_name

FROM departments e

LEFT OUTHER JOIN employees d ON (e.department_id=d.department_id);

b) dept de var empt de olmayanlar da dahil

SELECT e.last_name, d.ddepartment_id, d.department_name

FROM employees e

RİGHT OUTHER JOIN departmens d ON (e.department_id=d.department_id);

---FULL OUTHER---

Hem sağ hem sol söz konusu

c) Her iki tabloda da fazla olanlar gelir:

SELECT e.last_name, d.ddepartment_id, d.department_name

FROM employees e

FULL OUTHER JOIN departmens d ON (e.department_id=d.department_id);

Senario

Çalışanların eski işlerini göster?

SELECT last_name, e.job_id as "JOB", jb.job_id as "OLD JOB", end_date

FROM employees e LEFT OUTHER JOIN job_history jh

ON (e.employee_i = jh.employee_id);

NOTLAR:

//King iş değiştirmemiş.

//Kochar iş değiştirmiş.

4) SELF JOINS

Bir kayıtta bulunmayan bir değere daha ulaşmam gerekirse SELF JOIN kullanırız.

Ahmet müdür 101

Ali çalışan 202 müdürünün kodu 101 ---> employees

202 id li çalışanı getir, yanında müdürünün adını da getir. Bunun için self-joın gerekir.

Çalışan kodu 101 olan kim? Önce bunun cevabını bulmam gerekir.

SELECT worker.last_name || 'works for' || manager.last_name as "WORKS FOR"

FROM employees worker JOIN employees manager (2 tane ayrı tablo olusturduk zanneder sql)

ON (worker.manager_id = manager.employee_id);

SELECT worker.last_name || 'works for' || manager_id as "WORKS FOR"

FROM employees

NOT: manager id olsaydı eşleştirmeye ve joın'e ihtiyaç kalmazdı.

```
SELECT worker.lastname, worker.manager_id, manager.last_name as "manager name"

FROM employees worker JOIN employees manager

ON ()
```

NOT: ON kısmında eşleştirmeyı doğru yapmak asıl meseledir. En çok hata bu kısımda meydana gelir.