

Oracle SQL Eğitim Projesi

Barış Tansel Şeneş

18 Eylül 2025

Kullanılan veri Modeli : Oracle Live SQL
sitesindeki hazır Human Resources(HR)

Barissns@hotmail.com
Tanselsenes@gmail.com

GitHub : <https://github.com/Barissns>

LinkedIn : <https://www.linkedin.com/in/baristsenes>

GİRİŞ:

Bu proje, Oracle SQL eğitiminde edinilen bilgilerin uygulamalı olarak pekiştirilmesini amaçlamaktadır. Eğitim süresince SQL'ın temel yapıları, fonksiyon kullanımı, çoklu tablo işlemleri, alt sorgular, analitik fonksiyonlar ve Oracle'a özgü komutlar gibi konular detaylı şekilde ele alınmıştır.

Proje kapsamında Oracle Live SQL platformu kullanılarak HR (Human Resources) veri tabanı üzerinde çeşitli sorgular geliştirilmiştir. Bu veri tabanı, çalışanlar, departmanlar, lokasyonlar ve görevler gibi kurumsal yapıyı temsil eden tablolar içерdiği için SQL uygulamaları açısından oldukça uygun bir örnek teşkil etmektedir.

Hazırlanan sorgular; veri filtreleme, sıralama, gruptlama, tablo birleştirme (JOIN), alt sorgu kullanımı, analitik sıralama ve Oracle'a özgü fonksiyonların uygulanması gibi farklı teknikleri içermektedir. Böylece hem teorik bilgilerin pratiğe dökülmesi hem de SQL dilinin farklı yönlerinin anlaşılması hedeflenmiştir. Bu raporda, kullanılan veri modeli, geliştirilen sorgular ve elde edilen çıktılar detaylı şekilde açıklanmıştır. Proje sonunda hem SQL diline olan hakimiyetin artması hem de gerçek veri üzerinde analiz yapma becerisinin gelişmesi amaçlanmıştır.

Veri Modeli Açıklaması

HR veri tabanı şu tabloları içerir:

****HR.EMPLOYEES** - Çalışan bilgileri**

- EMPLOYEE_ID (Çalışan ID)
- FIRST_NAME (Ad)
- LAST_NAME (Soyad)
- EMAIL (E-posta)
- PHONE_NUMBER (Telefon)
- HIRE_DATE (İşe başlama tarihi)
- JOB_ID (İş pozisyonu ID)
- SALARY (Maaş)
- COMMISSION_PCT (Komisyon oranı)
- MANAGER_ID (Yönetici ID)
- DEPARTMENT_ID (Departman ID)

****HR.DEPARTMENTS** - Departman bilgileri**

- DEPARTMENT_ID (Departman ID)
- DEPARTMENT_NAME (Departman adı)
- MANAGER_ID (Yönetici ID)
- LOCATION_ID (Lokasyon ID)

****HR.JOBS** - İş pozisyonları**

- JOB_ID (İş ID)
- JOB_TITLE (İş unvanı)
- MIN_SALARY (Minimum maaş)
- MAX_SALARY (Maksimum maaş)

****HR.LOCATIONS** - Lokasyon bilgileri**

- LOCATION_ID (Lokasyon ID)
- STREET_ADDRESS (Adres)
- POSTAL_CODE (Posta kodu)
- CITY (Şehir)
- STATE_PROVINCE (Eyalet/il)
- COUNTRY_ID (Ülke ID)

****HR.COUNTRIES** - Ülke bilgileri**

- COUNTRY_ID (Ülke ID)
- COUNTRY_NAME (Ülke adı)
- REGION_ID (Bölge ID)

****HR.REGIONES** - Bölge bilgileri**

- REGION_ID (Bölge ID)
- REGION_NAME (Bölge adı)

****HR.JOB_HISTORY** - İş geçmişi**

- EMPLOYEE_ID (Çalışan ID)
- START_DATE (Başlangıç tarihi)
- END_DATE (Bitiş tarihi)
- JOB_ID (İş ID)
- DEPARTMENT_ID (Departman ID)

Kodlarım ve çıktıları Ektedir:

```
-- Bu sorgu, tüm çalışanların ad ve soyadlarını listeler.  
-- SELECT ile sütunları seçiyoruz, FROM ile tabloyu belirtiyoruz.  
SELECT first_name, last_name  
FROM hr.employees;
```

1	Ellen	Abel
2	Sundar	Ande
3	Mozhe	Atkinson
4	Shelli	Baida
5	Amit	Banda
6	Elizabeth	Bates
7	Sarah	Bell
8	David	Bernstein
9	Laura	Riccio

107 Adet Sonuç

```
-- Bu sorgu, maaşı 10.000'den fazla olan çalışanları getirir.  
-- WHERE ile filtreleme yapıyoruz, sadece salary > 10000 olanlar gelir.  
SELECT first_name, salary  
FROM hr.employees  
WHERE salary > 10000;
```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.004 seconds

	FIRST_NAME	SALARY	
1	Steven	24000	15 Adet Sonuç.
2	Neena	17000	
3	Lex	17000	

```
-- Bu sorgu, çalışanları işe giriş tarihine göre sıralar.
-- ORDER BY hire_date ASC → en eski işe girenler en üstte olur.
SELECT first_name, hire_date
FROM hr.employees
ORDER BY hire_date ASC;
```

	FIRST_NAME	HIRE_DATE
1	Lex	1/13/2011, 12:00:00
2	William	6/7/2012, 12:00:00
3	Hermann	6/7/2012, 12:00:00

-- Bu sorgu, maaşı 10.000'den fazla olan çalışanları maaşlarına göre azalan şekilde sıralar.
-- ORDER BY salary DESC → en yüksek maaş en üstte olur.

```
SELECT first_name, salary
FROM hr.employees
WHERE salary > 10000
ORDER BY salary DESC;
```

	FIRST_NAME	SALARY
1	Steven	24000
2	Neena	17000
3	Lex	17000

-- Bu sorgu, departman ID'si 20, 30 veya 40 olan çalışanları listeler.
-- WHERE IN (...) ifadesiyle birden fazla değeri filtreleyebiliriz.
-- ORDER BY department_id → departmanlara göre sıralama yapılır.

```
SELECT first_name, last_name, department_id
FROM hr.employees
WHERE department_id IN (20, 30, 40)
ORDER BY department_id
```

Clear output
Download
Execution time: 0.004 seconds

	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
1	Michael	Martinez	9 Adet Sonuç
2	Pat	Davis	20
3	Den	Li	30

----- Fonksiyonlar (String, Sayısal, Tarih) -----

-- Bu sorgu, çalışanların ad ve soyadlarını birleştirerek tam isim oluşturur.

-- || operatörü ile string birleştirme yapılır.

```
SELECT first_name || ' ' || last_name AS full_name  
FROM hr.employees;
```

	FULL_NAME
1	Ellen Abel
2	Sundar Ande
3	Mozhe Atkinson

107 Adet
Sonuç

-- Bu sorgu, CONCAT fonksiyonu ile ad ve soyadı birleştirir.

-- CONCAT sadece iki string alır, boşluk eklemek için ayrı bir CONCAT gerekebilir.

```
SELECT CONCAT(first_name, last_name)  
FROM hr.employees;
```

	CONCAT(FIRST_NAI
1	EllenAbel
2	SundarAnde
3	MozheAtkinson

107 Adet Sonuç.

-- Bu sorgu, çalışanların ad ve soyadlarını büyük harfe çevirir.

-- UPPER fonksiyonu string ifadeleri büyük harfe dönüştürür.

```
SELECT UPPER(first_name), UPPER(last_name)  
FROM hr.employees;
```

	UPPER(FIRST_NAM	UPPER(LAST_NAME
1	ELLEN	ABEL
2	SUNDAR	ANDE
3	MOZHE	ATKINSON

107 Adet Sonuç

-- Bu sorgu, maaşların 500'e bölümünden kalanı verir.

-- MOD fonksiyonu sayısal işlemlerde kalan hesaplamak için kullanılır.

```
SELECT MOD(salary, 500)  
FROM hr.employees;
```

	MOD(SALARY,500)
1	0
2	0
3	0
4	0
5	0
6	300
7	300
8	200
9	8
10	0
11	200
...	...

107 Adet
Sonuç

--- Bu soru, çalışanların maaşına %25 zam eklenmiş halini gösterir.

--- Sayısal işlemler doğrudan SELECT içinde yapılabilir.

```
SELECT salary, salary * 1.25 AS increased salary
```

```
FROM hr.employees;
```

	SALARY	INCREASED_SALAR
1	24000	30000
2	17000	21250
3	17000	21250
4	9000	11250
5	6000	7500
6	4800	6000
7	4800	6000
8	4200	5250
9	12008	15010
10	9000	11250
11	8200	10250

107 Adet
Sonuc.

— Bu soru, çalışanların işe giriş tarihinden bugüne kadar geçen gün sayısını hesaplar.

-- SYSDATE bugünün tarihini verir, DATE farkı gün cinsinden hesaplanır.

```
SELECT first_name, SYSDATE - hire_date AS days_worked
```

```
SELECT first_name  
FROM hr.employees;
```

**107 Adet
Sonuç**

-- Bu soru, 1999-01-01 tarihinden sonra işe başlayan çalışanları listeler.

-- DATE 'YYYY-MM-DD' formatı ile sabit tarih karşılaştırması yapılır.

SELECT first name, last name, hire date

```
SELECT first_name  
FROM hr.employees
```

```
WHERE hire_date > DATE '1999-01-01' ;
```

	FIRST_NAME	LAST_NAME	HIRE_DATE	
1	Steven	King	6/17/2013, 12:00:00 AM	
2	Neena	Yang	9/21/2015, 12:00:00 AM	
3	Lex	Garcia	1/13/2011, 12:00:00 AM	
4	Alexander	James	1/3/2016, 12:00:00 AM	
5	Bruce	Miller	5/21/2017, 12:00:00 AM	
6	David	Williams	6/25/2015, 12:00:00 AM	
7	Valli	Jackson	2/5/2016, 12:00:00 AM	
8	Diana	Nguyen	2/7/2017, 12:00:00 AM	
9	Nancy	Gruenberg	8/17/2012, 12:00:00 AM	
10	Daniel	Faviet	8/16/2012, 12:00:00 AM	
11	John	Chen	9/28/2015, 12:00:00 AM	
12				107 Adet Sonuç

**107 Adet
Sonuç**

```
-- Bu sorgu, bugünün tarihini DD-MM-YYYY formatında gösterir.
-- DUAL tablosu Oracle'da tek satırlık işlemler için kullanılır.
SELECT TO_CHAR(SYSDATE, 'DD-MM-YYYY') AS today_date
FROM DUAL;
```

TODAY_DATE	
1	18-09-2025

----- GROUP BY, HAVING ve Toplama Fonksiyonları -----

```
-- Bu sorgu, her departmanın ortalama maaşını hesaplar.
-- AVG(salary) ile ortalama alınır, GROUP BY ile departman bazında gruptama yapılır.
SELECT department_id, AVG(salary) AS avg_salary
FROM hr.employees
GROUP BY department_id;
```

Query result			
	DEPARTMENT_ID	AVG_SALARY	
1	90	19333.33333333332	
2	60	5760	
3	100	8601.33333333334	
4	30	4150	
5	50	3475.5555555555557	
6	80	8955.882352941177	
7	(null)	7000	
8	10	4400	
9	20	9500	
10	40	6500	
11	70	10000	
12	110	10154	

```
-- Bu sorgu, her departmanda kaç çalışan olduğunu gösterir.
-- COUNT(*) ile toplam kişi sayısı alınır, GROUP BY ile departmanlara göre ayrılır.
SELECT department_id, COUNT(*) AS employee_count
FROM hr.employees
GROUP BY department_id;
```

	DEPARTMENT_ID	EMPLOYEE_COUNT
1	90	3
2	60	5
3	100	6
4	30	6
5	50	45
6	80	34
7	(null)	1
8	10	1
9	20	2
10	40	1
11	70	1
12	110	2

-- Bu sorgu, sadece 8'den fazla çalışanı olan departmanları listeler.

-- HAVING COUNT(*) > 8 → grupta sonrası filtreleme yapılır.

```
SELECT department_id, COUNT(*) AS employee_count  
FROM hr.employees  
GROUP BY department_id  
HAVING COUNT(*) > 8;
```

Execution Time: 0 seconds

	DEPARTMENT_ID	EMPLOYEE_COUNT
1	50	45
2	80	34

-- Bu sorgu, ortalama maaşı 777'den büyük olan iş pozisyonlarını getirir.

-- GROUP BY job_id → pozisyonna göre grupta, HAVING ile filtreleme yapılır.

```
SELECT job_id, AVG(salary) AS avg_salary  
FROM hr.employees  
GROUP BY job_id  
HAVING AVG(salary) > 777;
```

	JOB_ID	Avg_Salary
9	ST_CLERK	2785
10	SA_MAN	12200
11	SA_REP	8350
12	SH_CLERK	3215
13	AD_ASST	4400
14	MK_MAN	13000
15	MK_REP	6000
16	HR_REP	6500
17	PR_REP	10000
18	AC_MGR	12008
19	AC_ACCOUNT	8300

-- Bu sorgu, yıl ve departmana göre işe alım sayısını gösterir.

-- EXTRACT(year from hire_date) → işe alım yılını çıkarır.

-- GROUP BY ile yıl ve departman bazında grupta yapılır.

-- ORDER BY ile sıralama yapılır.

```
SELECT EXTRACT(YEAR FROM hire_date) AS hire_year,  
       department_id,  
       COUNT(*) AS hires  
FROM hr.employees  
GROUP BY EXTRACT(YEAR FROM hire_date), department_id  
ORDER BY hire_year, department_id;
```

	HIRE_YEAR	DEPARTMENT_ID	Hires
27	2017	50	9
28	2017	60	2
29	2017	80	5
30	2017	100	1
31	2017	(null)	1
32	2018	50	4
33	2018	80	7

```
-- Bu sorgu, İsveç'te çalışan personellerin ad, soyad ve şehir bilgilerini listeler.
-- employees → departments → locations → countries tabloları INNER JOIN ile bağlanır.
-- WHERE ile sadece country_name = 'Sweden' olanlar filtrelenir.
SELECT e.first_name, e.last_name, l.city
FROM hr.employees e
INNER JOIN hr.departments d ON e.department_id = d.department_id
INNER JOIN hr.locations l ON d.location_id = l.location_id
INNER JOIN hr.countries c ON l.country_id = c.country_id
WHERE c.country_name = 'Sweden';
```

FIRST_NAME	LAST_NAME	CITY
------------	-----------	------

No items to display.

```
-- Bu sorgu, maaşı 7750'den fazla olan çalışanların ad, soyad, maaş ve iş unvanlarını gösterir.
-- employees tablosu jobs tablosuya INNER JOIN ile bağlanır.
```

```
SELECT e.first_name, e.last_name, e.salary, j.job_title
FROM hr.employees e
INNER JOIN hr.jobs j ON e.job_id = j.job_id
WHERE e.salary > 7750;
```

	FIRST_NAME	LAST_NAME	SALARY	JOB_TITLE
33	David	Bernstein	9500	Sales Representative
34	Sean	Tucker	10000	Sales Representative
35	Allan	McEwen	9000	Sales Representative
36	Payam	Kaufling	7900	Stock Manager
37	Adam	Fripp	7900	Stock Manager
38	Matthew	Weiss	8000	Stock Manager

```
-- Bu sorgu, hiç çalışmayı olmayan departmanları listeler.
```

```
-- LEFT JOIN ile tüm departmanlar alınır, çalışmayı olmayanlar NULL olur.
```

```
-- WHERE e.employee_id IS NULL → çalışmayı olmayanları filtreler.
```

```
SELECT d.department_name
FROM hr.departments d
LEFT JOIN hr.employees e ON d.department_id = e.department_id
WHERE e.employee_id IS NULL;
```

	DEPARTMENT_NAME
11	NOC
12	IT Helpdesk
13	Government Sales
14	Retail Sales
15	Recruiting
16	Payroll

```
-- Bu sorgu, çalışanları ve departman adlarını RIGHT JOIN ile eşleştirir.
```

```
-- RIGHT JOIN → tüm departmanlar alınır, çalışmayı olmayanlar da görünür.
```

```
SELECT e.first_name, e.last_name, d.department_name
```

```
FROM hr.employees e
RIGHT JOIN hr.departments d ON e.department_id = d.department_id;
```

	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
117	(null)	(null)	NOC
118	(null)	(null)	IT Helpdesk
119	(null)	(null)	Government Sales
120	(null)	(null)	Retail Sales
121	(null)	(null)	Recruiting
122	(null)	(null)	Payroll

```
-- Bu sorgu, çalışanlar ve departmanlar arasında FULL OUTER JOIN yapar.  
-- FULL JOIN → hem çalışanı olmayan departmanlar hem de departmanı olmayan çalışanlar görünür.  
155  
156 SELECT e.first_name, e.last_name, d.department_name  
157 FROM hr.employees e  
158 FULL OUTER JOIN hr.departments d ON e.department_id = d.department_id;
```

	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
118	(null)	(null)	Treasury
119	(null)	(null)	Manufacturing
120	(null)	(null)	Corporate Tax
121	(null)	(null)	IT Helpdesk
122	(null)	(null)	Shareholder Services
123	(null)	(null)	Benefits

```
159  
160 -- Bu sorgu, CROSS JOIN ile her çalışanı her departmanla eşleştirir.  
161 -- CROSS JOIN → kartesyen çarpım, toplam satır sayısı = çalışan × departman sayısı.  
162 SELECT e.first_name, e.last_name, d.department_name  
163 FROM hr.employees e  
164 CROSS JOIN hr.departments d;
```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.009 seconds

	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
216	Sundar	Ande	Purchasing
217	Mozhe	Atkinson	Purchasing
218	Shelli	Baida	Purchasing
219	Amit	Banda	Purchasing
220	Elizabeth	Bates	Purchasing
221	Sarah	Bell	Purchasing

```
165 -- Bu sorgu, çalışanların yöneticilerini gösterir.  
166 -- SELF JOIN → aynı tablo kendisiyle bağlanır.  
167 -- manager_id → employee_id eşleştirilerek çalışan ve yönetici eşleştirilir.  
168 SELECT e.first_name AS employee_name,  
169 ..... m.first_name AS manager_name  
170 FROM hr.employees e  
171 LEFT JOIN hr.employees m ON e.manager_id = m.employee_id;
```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.006 seconds

	EMPLOYEE_NAME	MANAGER_NAME
102	Alyssa	Eleni
103	Jonathon	Eleni
104	Jack	Eleni
105	Kimberely	Eleni
106	Charles	Eleni
107	Steven	(null)

```

170 ----- Alt Sorgular (Subqueries) -----
171
172 -- Bu sorgu, en yüksek maaşı alan çalışanı getirir.
173 -- İçteki alt sorgu MAX(salary) ile en yüksek maaşı bulur.
174 -- Dış sorgu, bu maaşa sahip olan çalışanı listeler.
175 SELECT first_name, salary
176 FROM hr.employees
177 WHERE salary = (SELECT MAX(salary) FROM hr.employees);

```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.004 seconds

	FIRST_NAME	SALARY
1	Steven	24000

```

178
179 -- Bu sorgu, ortalama maaştan fazla kazanan çalışanları listeler.
180 -- Alt sorgu AVG(salary) ile ortalama maaşı hesaplar.
181 -- Dış sorgu, bu değerden yüksek maaş alanları filtreler.
182 SELECT first_name, salary
183 FROM hr.employees
184 WHERE salary > (SELECT AVG(salary) FROM hr.employees);
185
186

```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.006 seconds

	FIRST_NAME	SALARY
46	Kimberely	7000
47	Michael	13000
48	Susan	6500
49	Hermann	10000
50	Shelley	10008
51	William	8300

```

186 -- Bu sorgu, IT departmanındaki çalışanları listeler.
187 -- Alt sorgu, 'IT' adlı departmanın ID'sini bulur.
188 -- Dış sorgu, bu ID'ye sahip çalışanları getirir.
189 SELECT first_name
190 FROM hr.employees
191 WHERE department_id =
192 |   SELECT department_id FROM hr.departments WHERE department_name = 'IT'
193 )|;
194

```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.005 seconds

	FIRST_NAME
1	Alexander
2	Bruce
3	David
4	Valli
5	Diana

```

195 -- Bu sorgu, departman ID'si 10 olan birimin ortalaması maaşından fazla kazanan çalışanları listeler.
196 -- Alt sorgu, sadece departman 10 için AVG(salary) hesaplar.
197 -- Dış sorgu, bu değerden yüksek maaş alanları filtreler.
198 SELECT first_name, last_name, salary, department_id
199 FROM hr.employees
200 WHERE salary > (
201     SELECT AVG(salary)
202     FROM hr.employees
203     WHERE department_id = 10
204 );
205

```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.006 seconds

	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_ID
55	Michael	Martinez	13000	20
56	Pat	Davis	6000	20
57	Susan	Jacobs	6500	40
58	Hermann	Brown	10000	70
59	Shelley	Higgins	12008	110
60	William	Gietz	8300	110

```

206 ----- UNION – UNION ALL, CTE (WITH) -----
207
208 -- Bu sorgu, IT (60) ve Finance (100) departmanlarındaki çalışanları birleştirir.
209 -- UNION → aynı sütun yapısına sahip iki sorgunun sonucunu birleştirir, tekrarları kaldırır.
210 SELECT first_name
211 FROM hr.employees
212 WHERE department_id = 60
213 UNION
214 SELECT first_name
215 FROM hr.employees
216 WHERE department_id = 100;

```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.005 seconds

	FIRST_NAME
6	Nancy
7	Daniel
8	John
9	Ismael
10	Jose Manuel
11	Luis

```

218 -- Bu sorgu, maaşı 10.000'den fazla olan çalışanlar ile yöneticileri tek listede birleştirir.
219 -- UNION ile iki farklı sorgu birleştirilir, 'Category' sütunu ile gruplar etiketlenir.
220 -- Alt sorgu ile yöneticilerin ID'leri alınır.
221 SELECT first_name, last_name, 'Yüksek Maaşlı' AS Category, salary
222 FROM hr.employees
223 WHERE salary > 10000
224 UNION
225 SELECT first_name, last_name, 'Manager' AS Category, salary
226 FROM hr.employees
227 WHERE employee_id IN (
228     SELECT DISTINCT manager_id
229     FROM hr.employees
230     WHERE manager_id IS NOT NULL
231 );
232
233 -- Bu sorgu, IT ve Finance departmanlarındaki çalışanları tekrarları da dahil ederek birleştirir

```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.007 seconds

	FIRST_NAME	LAST_NAME	CATEGORY	SALARY
28	Karen	Partners	Manager	13500
29	Alberto	Errazuriz	Manager	12000
30	Gerald	Cambrault	Manager	11000
31	Eleni	Zlotkey	Cambrault Manager	10500
32	Michael	Martinez	Manager	13000
33	Shelley	Higgins	Manager	12008

```

233 -- Bu sorgu, IT ve Finance departmanlarındaki çalışanları tekrarları da dahil ederek birleştirir.
234 -- UNION ALL → UNION'dan farklı olarak tekrarları kaldırır.
235 SELECT first_name
236 FROM hr.employees
237 WHERE department_id = 60
238 UNION ALL
239 SELECT first_name
240 FROM hr.employees
241 WHERE department_id = 100;

```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.004 seconds

	FIRST_NAME
6	Nancy
7	Daniel
8	John
9	Ismael
10	Jose Manuel
11	Luis

```

243 -- Bu sorgu, CTE (Common Table Expression) kullanarak yüksek maaşlı çalışanları departman bazında sayar.
244 -- WITH ile geçici bir tablo (HIGH_EARNERS) tanımlanır.
245 -- Bu tablo JOIN ile departmanlara bağlanır ve COUNT ile kişi sayısı hesaplanır.
246 WITH HIGH_EARNERS AS (
247   SELECT first_name, last_name, salary, department_id
248   FROM hr.employees
249   WHERE salary > (SELECT AVG(salary) FROM hr.employees)
250 )
251 SELECT d.department_name, COUNT(*) AS high_earner_count
252 FROM HIGH_EARNERS h
253 JOIN hr.departments d ON h.department_id = d.department_id
254 GROUP BY d.department_name
255 ORDER BY high_earner_count DESC;

```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.011 seconds

	DEPARTMENT_NAME	HIGH_EARNER_COU
5	Accounting	2
6	Public Relations	1
7	Human Resources	1
8	Purchasing	1
9	Marketing	1
10	IT	1

```

258 -- 1. CTE: DEPT_STATS → departman bazında ortalama maaş ve çalışan sayısı
259 -- 2. CTE: HIGH_SALARY_DEPTS → ortalama maaşı 8000'den fazla olan departmanlar
260 -- 3. CTE: LOCATION_INFO → departmanların şehir ve Ülke bilgileri
261 -- Son sorgu: çalışanlar, yüksek maaşlı departmanlar ve lokasyon bilgileri birleştirilir.
262 WITH
263 DEPT_STATS AS (
264   SELECT department_id, AVG(salary) AS avg_salary, COUNT(*) AS emp_count
265   FROM hr.employees
266   GROUP BY department_id
267 ),
268 HIGH_SALARY_DEPTS AS (
269   SELECT department_id
270   FROM DEPT_STATS
271   WHERE avg_salary > 8000
272 ),
273 LOCATION_INFO AS (
274   SELECT d.department_id, d.department_name, l.city, c.country_name
275   FROM hr.departments d
276   JOIN hr.locations l ON d.location_id = l.location_id
277   JOIN hr.countries c ON l.country_id = c.country_id
278 )
279 SELECT e.first_name, e.last_name, e.salary, li.department_name, li.city, li.country_name
280 FROM hr.employees e
281 JOIN HIGH_SALARY_DEPTS hsd ON e.department_id = hsd.department_id
282 JOIN LOCATION_INFO li ON e.department_id = li.department_id
283 ORDER BY e.salary DESC;
284

```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.017 seconds

	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_NAME	CITY	COUNTRY_NAME
47	Sundita	Kumar	6100	Sales	Oxford	United Kingdom of G
48	Pat	Davis	6000	Marketing	Toronto	Canada

```

285 ----- Analitik Fonksiyonlar (ROW_NUMBER, RANK) -----
286
287 -- Bu sorgu, çalışanlara maaşlarına göre sıra numarası verir.
288 -- ROW_NUMBER() → her satırda benzersiz bir sıra numarası atar.
289 -- ORDER BY salary DESC → en yüksek maaş en üstte olur.
290 SELECT first_name, salary,
291     .... ROW_NUMBER() OVER (ORDER BY salary DESC) AS row_num
292 FROM hr.employees;
293

```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.005 seconds

	FIRST_NAME	SALARY	ROW_NUM
106	Steven	2200	106
107	TJ	2100	107

294 -- Bu sorgu, aynı maaşa sahip çalışanlara aynı sıralama değerini verir. 295 -- RANK() → eşit değerlere aynı sıralama numarasını verir, sonraki sıra atlanır. 296 SELECT first_name, salary, 297 RANK() OVER (ORDER BY salary DESC) AS salary_rank 298 FROM hr.employees;			

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.005 seconds

	FIRST_NAME	SALARY	SALARY_RANK
106	Steven	2200	105
107	TJ	2100	107

300 -- Bu sorgu, her departman içinde maaş sıralaması yapar. 301 -- PARTITION BY department_id → her departman ayrı bir grup gibi değerlendirilir. 302 -- RANK() → departman içindeki maaş sıralamasını verir. 303 SELECT first_name, department_id, salary, 304 RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS dept_rank 305 FROM hr.employees; 306			

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.005 seconds

	FIRST_NAME	DEPARTMENT_ID	SALARY	DEPT_RANK
104	Luis	100	6900	0
105	Shelley	110	12008	1
106	William	110	8300	2
107	Kimberely	(null)	7000	1

```

307 -- Bu soru, maaş dağılımında en üst %10'luk dilimde yer alan çalışanları analiz eder.
308 -- 1. CTE: SALARY_PERCENTILES → PERCENT_RANK ile maaş yüzdelik dilimi hesaplanır.
309 -- 2. CTE: TOP_10_PERCENT → %10'dan yüksek olanlar filtrelenir.
310 -- Son soru: çalışan bilgileri, departman, iş unvanı, şehir, ülke, deneyim süresi, komisyon durumu ve yönetici kişi sayısı gösterilir.
311 WITH SALARY_PERCENTILES AS (
312   SELECT employee_id, first_name, last_name, salary,
313         PERCENT_RANK() OVER (ORDER BY salary DESC) AS salary_percentile
314   FROM hr.employees
315 ),
316 TOP_10_PERCENT AS (
317   SELECT * FROM SALARY_PERCENTILES WHERE salary_percentile <= 0.1
318 )
319 SELECT t.first_name, t.last_name, t.salary, d.department_name, j.job_title,
320       l.city, c.country_name,
321       ROUND(MONTHS_BETWEEN(SYSDATE, e.hire_date)/12, 1) AS years_experience,
322       CASE WHEN e.commission_pct IS NOT NULL THEN 'Has Commission' ELSE 'No Commission' END AS commission_status,
323       (SELECT COUNT(*) FROM hr.employees e2 WHERE e2.manager_id = e.employee_id) AS direct_reports
324 FROM TOP_10_PERCENT t
325 JOIN hr.employees e ON t.employee_id = e.employee_id
326 JOIN hr.departments d ON e.department_id = d.department_id
327 JOIN hr.jobs j ON e.job_id = j.job_id
328 JOIN hr.locations l ON d.location_id = l.location_id
329 JOIN hr.countries c ON l.country_id = c.country_id
330 ORDER BY t.salary DESC;
331

```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.016 seconds

	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_NAME	JOB_TITLE	CITY	COUNTRY_NAME	YEARS_EXPERIENCE	COMMISSION_STATUS	DIRECT_REPORTS
10	Lisa	Ozer	11500	Sales	Sales Representative	Oxford	United Kingdom of G	10.5	Has Commission	0
11	Gerald	Cambrault	11000	Sales	Sales Manager	Oxford	United Kingdom of G	7.9	Has Commission	6
12	Ellen	Abel	11000	Sales	Sales Representative	Oxford	United Kingdom of G	11.4	Has Commission	0
13	Den	Li	11000	Purchasing	Purchasing Manager	Seattle	United States of Ame	12.8	No Commission	5

```

333 ----- Oracle'a Özgü Fonksiyonlar (NVL, DECODE, DUAL, CONNECT BY) -----
334
335 -- Bu soru, çalışanların komisyon bilgilerini gösterir.
336 -- NVL fonksiyonu, NULL olan commission_pct değerlerini 0 ile değiştirir.
337 -- Oracle'a özgü bir fonksiyondur, veri eksikliğini yönetmek için kullanılır.
338 SELECT first_name, NVL(commission_pct, 0) AS commissions
339 FROM hr.employees;

```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.006 seconds

	FIRST_NAME	COMMISSIONS
104	Susan	0
105	Hermann	0
106	Shelley	0
107	William	0

```

341 -- Bu soru da aynı şekilde, komisyonu olmayanlara "0" değeri verir.
342 -- NVL(commission_pct, 0) → NULL olanları 0 yapar.
343 SELECT first_name, NVL(commission_pct, 0) AS commission
344 FROM hr.employees;
345
346 -- Bu soru, NVL2 fonksiyonu ile komisyon durumu hakkında metinsel bilgi

```

Query result Script output DBMS output Explain Plan SQL history

Download ▾ Execution time: 0.005 seconds

	FIRST_NAME	COMMISSION
104	Susan	0
105	Hermann	0
106	Shelley	Hermann
107	William	0

```

346 -- Bu sorgu, NVL2 fonksiyonu ile komisyon durumu hakkında metinsel bilgi verir.
347 -- NVL2 → NULL değilse 'yes commission', NULL ise 'no commission' döner.
348 SELECT first_name, NVL2(commission_pct, 'yes commission', 'no commission') AS commission_status
349 FROM hr.employees;
350
351 -- Bu sorgu, çalışanların iş tanımına göre rol belirler.

```

Query result Script output DBMS output Explain Plan SQL history

trash info Download Execution time: 0.005 seconds

	FIRST_NAME	COMMISSION_STAT
104	Susan	no commission
105	Hermann	no commission
106	Shelley	no commission
107	William	no commission

```

351 -- Bu sorgu, çalışanların iş tanımına göre rol belirler.
352 -- DECODE → Oracle'a özgü koşullu dönüşüm fonksiyonudur.
353 -- Belirli job_id değerlerine karşılık gelen metinler döner.
354 SELECT job_id,
355     DECODE(job_id, 'IT_PROG', 'Developer', 'AD_VP', 'Vice President', 'Other') AS role
356 FROM hr.employees;
357

```

Query result Script output DBMS output Explain Plan SQL history

trash info Download Execution time: 0.005 seconds

	JOB_ID	ROLE
104	ST_MAN	Other
105	ST_MAN	Other
106	ST_MAN	Other
107	ST_MAN	Other

```

358 -- Bu sorgu, CONNECT BY ile çalışan-yönetici hiyerarşisini gösterir.
359 -- START WITH → en üst düzeyden (yönetici olmayan) başlar.
360 -- CONNECT BY PRIOR → kendini yöneticisiyle bağlar.
361 -- Oracle'in hiyerarşik sorgular için sunduğu özel bir yapıdır.
362 SELECT employee_id, first_name, manager_id
363 FROM hr.employees
364 START WITH manager_id IS NULL
365 CONNECT BY PRIOR employee_id = manager_id;

```

Query result Script output DBMS output Explain Plan SQL history

trash info Download Execution time: 0.006 seconds

	EMPLOYEE_ID	FIRST_NAME	MANAGER_ID
104	178	Kimbereiy	149
105	179	Charles	149
106	201	Michael	100
107	202	Pat	201

Sonuç ve Değerlendirme

Bu proje kapsamında Oracle SQL eğitiminde öğrenilen konuların büyük bir kısmı uygulamalı olarak test edilmiştir. HR veri tabanı üzerinde gerçekleştirilen sorgular sayesinde hem temel SQL komutları hem de ileri düzey fonksiyonlar hakkında pratik kazanılmıştır.

JOIN işlemleriyle tablolar arası ilişkileri kurmak, gruplama ve filtreleme ile veri analizi yapmak, alt sorgular ve analitik fonksiyonlarla daha karmaşık sorgular geliştirmek bu sürecin en öğretici adımları olmuştur. Özellikle ROW_NUMBER, RANK gibi analitik fonksiyonların kullanımını öğrenmek, gerçek hayatı sıralama ve raporlama ihtiyaçlarını karşılayabilecek çözümler üretmemi sağladı.

Oracle'a özgü fonksiyonlar olan NVL, DECODE ve CONNECT BY gibi yapılar sayesinde Oracle SQL'in sunduğu özel imkanları tanımiş oldum. Bu fonksiyonlar, veri eksikliği, koşullu dönüşüm ve hiyerarsık yapılarının modellenmesi gibi konularda oldukça faydalı oldu.

Genel olarak bu proje, SQL diline olan hakkimiyetimi artırılmış, veri modelleme ve sorgulama becerilerimi geliştirmiştir. Oracle Live SQL ortamı sayesinde sorguların çıktısını anlık olarak görerek deneme-yanılma yöntemiyle öğrenme sürecim hızlanmıştır. Bu deneyim, ileride daha büyük veri tabanlarıyla çalışırken karşılaşabileceğim senaryolara hazırlık niteliği taşımaktadır.