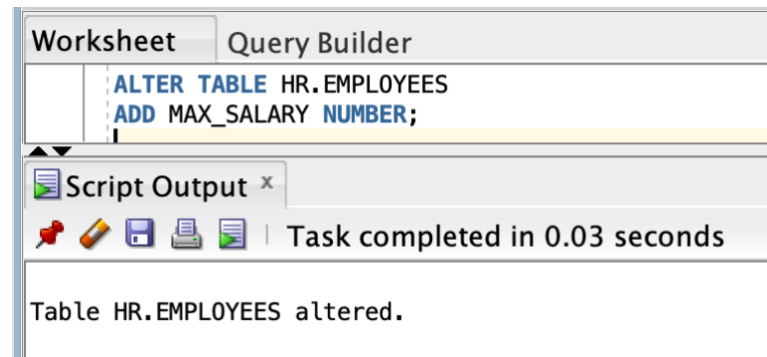


## Exercise SQL03-EX-01:

**Definiton :** Write followig SQL queries:

- Add a colum to employees table named MAX\_SALARY.
- Update MAX\_SALARY with maximum salary amount with subquery.
- Delete employee who have minimum salary using subquery.



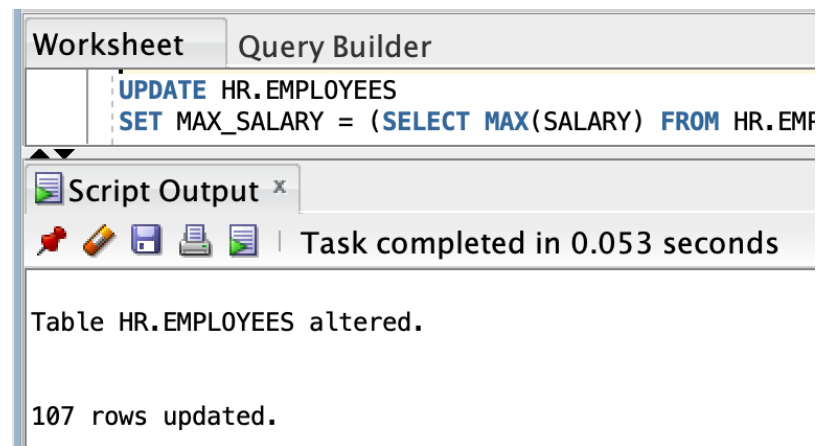
Worksheet Query Builder

```
ALTER TABLE HR.EMPLOYEES  
ADD MAX_SALARY NUMBER;
```

Script Output x

Task completed in 0.03 seconds

Table HR.EMPLOYEES altered.



Worksheet Query Builder

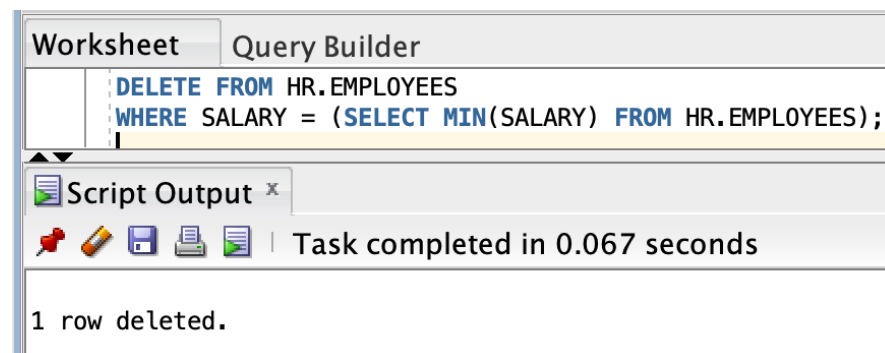
```
UPDATE HR.EMPLOYEES  
SET MAX_SALARY = (SELECT MAX(SALARY) FROM HR.EMPLOYEES);
```

Script Output x

Task completed in 0.053 seconds

Table HR.EMPLOYEES altered.

107 rows updated.



Worksheet Query Builder

```
DELETE FROM HR.EMPLOYEES  
WHERE SALARY = (SELECT MIN(SALARY) FROM HR.EMPLOYEES);
```

Script Output x

Task completed in 0.067 seconds

1 row deleted.

## Exercise SQL03-EX-02:

**Definiton :** Write followig SQL queries:

- Define index (named DPR\_NAME\_IDX) on DEPARTMENT\_NAME column of DEPARTMENTS table.
- Define constraint (named CNSTR\_SALARY) on employee salary. (Salary must be between 1000\$ and 100.000\$)
- Drop defined index.
- Enable, disable, drop defined constraint.

**Worksheet** **Query Builder**

```
CREATE INDEX DPR_NAME_IDX
ON HR.DEPARTMENTS (DEPARTMENT_NAME);
```

**Script Output** x

Task completed in 0.078 seconds

Index DPR\_NAME\_IDX created.

```
ALTER TABLE HR.EMPLOYEES
ADD CONSTRAINT CNSTR_SALARY
CHECK (SALARY BETWEEN 1000 AND 100000);
```

**Script Output** x

Task completed in 0.028 seconds

Table HR.EMPLOYEES altered.

```
DROP INDEX DPR_NAME_IDX;
```

**Script Output** x

Task completed in 0.174 seconds

Index DPR\_NAME\_IDX dropped.

```
ALTER TABLE HR.EMPLOYEES
DISABLE CONSTRAINT CNSTR_SALARY;
```

**Script Output** x

Task completed in 0.06 seconds

Table HR.EMPLOYEES altered.

```
ALTER TABLE HR.EMPLOYEES  
ENABLE CONSTRAINT CNSTR_SALARY;
```

Script Output x

Task completed in 0.052 seconds

Table HR.EMPLOYEES altered.

## Exercise SQL03-EX-03:

**Definition :** Create a table from EMPLOYEES with distinct department\_id column. Add department\_name to that table. With DEPARTMENTS table, update department\_name for included department\_ids and insert department\_id and department\_name values for not included rows. Use MERGE keyword.

```
CREATE TABLE EMP_DEPT AS  
SELECT DISTINCT DEPARTMENT_ID  
FROM HR.EMPLOYEES;  
  
ALTER TABLE EMP_DEPT  
ADD DEPARTMENT_NAME VARCHAR2(50);
```

Script Output x

Task completed in 0.259 seconds

Table EMP\_DEPT created.

Table EMP\_DEPT altered.

```
MERGE INTO EMP_DEPT target  
USING (  
    SELECT DEPARTMENT_ID, DEPARTMENT_NAME  
    FROM HR.DEPARTMENTS  
) source  
ON (target.DEPARTMENT_ID = source.DEPARTMENT_ID)  
WHEN MATCHED THEN  
    UPDATE SET target.DEPARTMENT_NAME = source.DEPARTMENT_NAME  
WHEN NOT MATCHED THEN  
    INSERT (DEPARTMENT_ID, DEPARTMENT_NAME)  
    VALUES (source.DEPARTMENT_ID, source.DEPARTMENT_NAME);
```

Script Output x

Task completed in 0.054 seconds

Table EMP\_DEPT altered.

27 rows merged.

## Exercise SQL03-EX-04:

**Definiton :** Using **WITH** keyword, do following jobs:

- Firstly select first\_name, last\_name, job\_id, department\_id from employees table whoes job\_id starts with 'S'.
- Additionally select job\_title and min-max salary amount.
- Add department\_name to that query.
- Lastly concat first\_name and last\_name with space as full\_name alias and list with other selected columns.

```
WITH Employee_Info AS (
    SELECT
        FIRST_NAME,
        LAST_NAME,
        JOB_ID,
        DEPARTMENT_ID
    FROM HR.EMPLOYEES WHERE JOB_ID LIKE 'S%'
),
Job_Info AS (
    SELECT
        JOB_ID,
        JOB_TITLE,
        MIN_SALARY,
        MAX_SALARY
    FROM HR.JOBS
),
Full_Employee_Info AS (
    SELECT
        E.FIRST_NAME,
        E.LAST_NAME,
        E.JOB_ID,
        E.DEPARTMENT_ID,
        J.JOB_TITLE,
        J.MIN_SALARY,
        J.MAX_SALARY,
        D.DEPARTMENT_NAME
    FROM Employee_Info E
    JOIN Job_Info J ON E.JOB_ID = J.JOB_ID
    JOIN HR.DEPARTMENTS D ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
)
SELECT
    FIRST_NAME || ' ' || LAST_NAME AS FULL_NAME,
    JOB_ID,
    JOB_TITLE,
    MIN_SALARY,
    MAX_SALARY,
    DEPARTMENT_NAME
FROM Full_Employee_Info ORDER BY FULL_NAME;
```

| Query Result x                         |          |                      |            |            |                 |  |
|--|----------|----------------------|------------|------------|-----------------|--|
| SQL   Fetched 50 rows in 0.042 seconds |          |                      |            |            |                 |  |
| FULL_NAME                              | JOB_ID   | JOB_TITLE            | MIN_SALARY | MAX_SALARY | DEPARTMENT_NAME |  |
| 1 Adam Fripp                           | ST_MAN   | Stock Manager        | 5500       | 8500       | Shipping        |  |
| 2 Alana Walsh                          | SH_CLERK | Shipping Clerk       | 2500       | 5500       | Shipping        |  |
| 3 Alberto Errazuriz                    | SA_MAN   | Sales Manager        | 10000      | 20080      | Sales           |  |
| 4 Alexis Bull                          | SH_CLERK | Shipping Clerk       | 2500       | 5500       | Shipping        |  |
| 5 Allan McEwen                         | SA_REP   | Sales Representative | 6000       | 12008      | Sales           |  |
| 6 Alyssa Hutton                        | SA_REP   | Sales Representative | 6000       | 12008      | Sales           |  |
| 7 Amit Banda                           | SA_REP   | Sales Representative | 6000       | 12008      | Sales           |  |
| 8 Anthony Cabrio                       | SH_CLERK | Shipping Clerk       | 2500       | 5500       | Shipping        |  |
| 9 Britney Everett                      | SH_CLERK | Shipping Clerk       | 2500       | 5500       | Shipping        |  |

## Exercise SQL03-EX-05:

**Definiton :** Search for COMMIT and ROLLBACK keywords and explain them.

COMMIT o anki transactionda yapılan değışiklikleri kalıcı hale getirir.

ROLLBACK bir transactionda henüz commit edilmemiş olan değışiklikleri geri alır.