

## EXERCISE 3 AND 4

### Functions, Operators and Group Functions

#### LIVESQL LINKS:

Exercise 3: <https://livesql.oracle.com/apex/livesql/s/paz0d2hebp9yrds3o7809dxia>

Exercise 4: <https://livesql.oracle.com/apex/livesql/s/paz0d2hehh2rojbp0m749ufoi>

#### Exercise 3

Aim: To understand types of function in SQL

1. Apply SQL queries to retrieve data by using all numeric functions.

The following tables represent the data shown in the screenshots:

ABS(FUND)
1220.9
3290.89
43387.76
2800.5
5480.67

CEIL(FUND)
1221
3291
43388
2801
5481

COS(FUND)
-.38134802614634659979227032355386235921
.07163395875426032221189689270966131118
-.71362378430130767838025315123066736147
-.22783212855234273951581062085803743073
-.16091317223132938941281600080932926875

EXP(DEPTNUM)
2.71828182845904523536028747135266249776
20.08553692318766774092852965458171789706
148.413159102576603421115580040552279624
54.59815003314423907811026120286087840308
148.413159102576603421115580040552279624

The following table summarizes the SQL queries and their results shown in the screenshots:

Statement	SQL Query	Result						
28	SELECT FLOOR(FUND) FROM PROJECT	<table border="1"> <thead> <tr> <th>FLOOR(FUND)</th> </tr> </thead> <tbody> <tr><td>1220</td></tr> <tr><td>3290</td></tr> <tr><td>43387</td></tr> <tr><td>2800</td></tr> <tr><td>5480</td></tr> </tbody> </table>	FLOOR(FUND)	1220	3290	43387	2800	5480
FLOOR(FUND)								
1220								
3290								
43387								
2800								
5480								
29	SELECT POWER(100, DEPTNUM) FROM PROJECT	<table border="1"> <thead> <tr> <th>POWER(100,DEPTNUM)</th> </tr> </thead> <tbody> <tr><td>100</td></tr> <tr><td>1000000</td></tr> <tr><td>10000000000</td></tr> <tr><td>1000000000</td></tr> <tr><td>10000000000</td></tr> </tbody> </table>	POWER(100,DEPTNUM)	100	1000000	10000000000	1000000000	10000000000
POWER(100,DEPTNUM)								
100								
1000000								
10000000000								
1000000000								
10000000000								
30	SELECT MOD(NUM, DEPTNUM) FROM PROJECT	<table border="1"> <thead> <tr> <th>MOD(NUM,DEPTNUM)</th> </tr> </thead> <tbody> <tr><td>0</td></tr> <tr><td>1</td></tr> <tr><td>3</td></tr> <tr><td>3</td></tr> <tr><td>0</td></tr> </tbody> </table>	MOD(NUM,DEPTNUM)	0	1	3	3	0
MOD(NUM,DEPTNUM)								
0								
1								
3								
3								
0								
31	SELECT ROUND(FUND, 1) FROM PROJECT	<table border="1"> <thead> <tr> <th>ROUND(FUND,1)</th> </tr> </thead> <tbody> <tr><td>1220.9</td></tr> <tr><td>3290.9</td></tr> <tr><td>43387.8</td></tr> <tr><td>2800.5</td></tr> <tr><td>5480.7</td></tr> </tbody> </table>	ROUND(FUND,1)	1220.9	3290.9	43387.8	2800.5	5480.7
ROUND(FUND,1)								
1220.9								
3290.9								
43387.8								
2800.5								
5480.7								
32	SELECT TRUNC(FUND, 1) FROM PROJECT	<table border="1"> <thead> <tr> <th>TRUNC(FUND,1)</th> </tr> </thead> <tbody> <tr><td>1220.9</td></tr> <tr><td>3290.8</td></tr> <tr><td>43387.7</td></tr> <tr><td>2800.5</td></tr> <tr><td>5480.6</td></tr> </tbody> </table>	TRUNC(FUND,1)	1220.9	3290.8	43387.7	2800.5	5480.6
TRUNC(FUND,1)								
1220.9								
3290.8								
43387.7								
2800.5								
5480.6								
33	SELECT SQRT(NUM) FROM PROJECT	<table border="1"> <thead> <tr> <th>SQRT(NUM)</th> </tr> </thead> <tbody> <tr><td>58.20652884342099299103554658006372936562</td></tr> <tr><td>44.10215414239989252817754216680080748746</td></tr> <tr><td>81.78019320104349080945257003214433602108</td></tr> <tr><td>49.223977896955869651391131177796054904288</td></tr> <tr><td>88.0056816347671952965775458025066552969</td></tr> </tbody> </table>	SQRT(NUM)	58.20652884342099299103554658006372936562	44.10215414239989252817754216680080748746	81.78019320104349080945257003214433602108	49.223977896955869651391131177796054904288	88.0056816347671952965775458025066552969
SQRT(NUM)								
58.20652884342099299103554658006372936562								
44.10215414239989252817754216680080748746								
81.78019320104349080945257003214433602108								
49.223977896955869651391131177796054904288								
88.0056816347671952965775458025066552969								

## 2. Apply SQL queries to retrieve data by using all character functions.

The following tables represent the data retrieved from the SQL queries shown in the screenshots.

**Statement 34: SELECT INITCAP(ADDRESS) FROM EMPLOYEE**

INITCAP(ADDRESS)
11 S 59 E. Salt Lake City, Ut
35 S 8 E. Salt Lake City, Ut
638 Voss, Houston, Tx
291 Berry, Bellaire, Tx
731 Fondren, Houston, Tx
975 Fire Oak, Humble, Tx
5631 Rice, Houston, Tx
450 Stone, Houston, Tx
3321 Castle, Spring, Tx
980 Dallas, Houston, Tx

10 rows selected.

**Statement 35: SELECT LOWER(ADDRESS) FROM EMPLOYEE**

LOWER(ADDRESS)
11 s 59 e. salt lake city, ut
35 s 8 e. salt lake city, ut
638 voss, houston, tx
291 berry, bellaire, tx
731 fondren, houston, tx
975 fire oak, humble, tx
5631 rice, houston, tx
450 stone, houston, tx
3321 castle, spring, tx
980 dallas, houston, tx

10 rows selected.

**Statement 36: SELECT UPPER(ADDRESS) FROM EMPLOYEE**

UPPER(ADDRESS)
11 S 59 E. SALT LAKE CITY, UT
35 S 8 E. SALT LAKE CITY, UT
638 VOSS, HOUSTON, TX
291 BERRY, BELLAIRE, TX
731 FONDREN, HOUSTON, TX
975 FIRE OAK, HUMBLE, TX
5631 RICE, HOUSTON, TX
450 STONE, HOUSTON, TX
3321 CASTLE, SPRING, TX
980 DALLAS, HOUSTON, TX

10 rows selected.

**Statement 37: SELECT RTRIM(ADDRESS, 'TX') FROM EMPLOYEE**

RTRIM(ADDRESS, TX)
11 S 59 E. SALT LAKE CITY, U
35 S 8 E. SALT LAKE CITY, U
638 VOSS, HOUSTON,
291 BERRY, BELLAIRE,
731 FONDREN, HOUSTON,
975 FIRE OAK, HUMBLE,
5631 RICE, HOUSTON,
450 STONE, HOUSTON,
3321 CASTLE, SPRING,
980 DALLAS, HOUSTON,

10 rows selected.

**Statement 38: SELECT LTRIM(ADDRESS, '33') FROM EMPLOYEE**

LTRIM(ADDRESS, '33')
11 S 59 E. SALT LAKE CITY, UT
5 S 8 E. SALT LAKE CITY, UT
638 VOSS, HOUSTON, TX
291 BERRY, BELLAIRE, TX
731 FONDREN, HOUSTON, TX
975 FIRE OAK, HUMBLE, TX
5631 RICE, HOUSTON, TX
450 STONE, HOUSTON, TX
21 CASTLE, SPRING, TX
980 DALLAS, HOUSTON, TX

10 rows selected.

**Statement 39: SELECT TRANSLATE(ADDRESS, 'TX', 'UT') FROM EMPLOYEE**

TRANSLATE(ADDRESS, TX, UT)
11 S 59 E. SALT LAKE CITY, UU
35 S 8 E. SALT LAKE CITY, UU
638 VOSS, HOUSTON, UT
291 BERRY, BELLAIRE, UT
731 FONDREN, HOUSTON, UT
975 FIRE OAK, HUMBLE, UT
5631 RICE, HOUSTON, UT
450 STONE, HOUSTON, UT
3321 CASULE, SPRING, UT
980 DALLAS, HOUSTON, UT

10 rows selected.

Live SQL

Feedback Help aathira.s2021@vitstudent.ac.in

My Scripts \ Script

Statement 40

SELECT REPLACE(ADDRESS, 'TX', 'UT') FROM EMPLOYEE

REPLACE(ADDRESS, 'TX', 'UT')

11 5 59 E. SALT LAKE CITY, UT

35 5 8 E. SALT LAKE CITY, UT

638 VOSS, HOUSTON, UT

291 BERRY, BELLAIRE, UT

731 FONDREN, HOUSTON, UT

975 FIRE OAK, HUMBLE, UT

5631 RICE, HOUSTON, UT

450 STONE, HOUSTON, UT

3321 CASTLE, SPRING, UT

980 DALLAS, HOUSTON, UT

Download CSV

10 rows selected.

Live SQL

Feedback Help aathira.s2021@vitstudent.ac.in

My Scripts \ Script

Statement 41

SELECT SUBSTR(LNAME, 1, 3) FROM EMPLOYEE

SUBSTR(LNAME,1,3)

GIL

PAN

WONG

WALL

SMI

NAR

ENG

BOR

ZEL

JAB

Download CSV

10 rows selected.

Live SQL

Feedback Help aathira.s2021@vitstudent.ac.in

My Scripts \ Script

Statement 42

SELECT LPAD(LNAME, 10, '-') FROM EMPLOYEE

LPAD(LNAME,10,'-')

---GILBERT

-----PAN

-----WONG

---WALLACE

-----SMITH

---NARAYAN

---ENGLISH

-----BORG

---ZELAYA

-----JABBAR

Download CSV

10 rows selected.

Live SQL

Feedback Help aathira.s2021@vitstudent.ac.in

My Scripts \ Script

Statement 43

SELECT RPAD(LNAME, 10, '-') FROM EMPLOYEE

RPAD(LNAME,10,'-')

GILBERT---

PAN-----

WONG-----

WALLACE---

SMITH-----

NARAYAN---

ENGLISH---

BORG-----

ZELAYA-----

JABBAR-----

Download CSV

10 rows selected.

Live SQL

Feedback Help aathira.s2021@vitstudent.ac.in

My Scripts \ Script

Statement 45

SELECT LENGTH(LNAME) FROM EMPLOYEE

LENGTH(LNAME)

7

3

4

7

5

7

7

4

6

6

Download CSV

10 rows selected.

Live SQL

Feedback Help aathira.s2021@vitstudent.ac.in

My Scripts \ Script

Statement 44

SELECT CHR(NUM+100) FROM DEPARTMENT

CHR(NUM+100)

e

f

g

h

i

Download CSV

5 rows selected.

### 3. Apply SQL queries to retrieve data by using all date functions.

The following tables represent the data retrieved from the SQL queries shown in the screenshots:

#### Statement 46: ADD\_MONTHS(BDAY, 2) FROM EMPLOYEE

ADD_MONTHS(BDAY,2)
09-AUG-60
07-APR-78
08-FEB-46
20-AUG-31
09-MAR-55
15-NOV-52
30-SEP-62
10-JAN-28
19-SEP-58
29-MAY-59

#### Statement 47: MONTHS\_BETWEEN(SYSDATE, BDAY) FROM EMPLOYEE

MONTHS_BETWEEN(SYSDATE, BDAY)
756
544.084324223416965352449223416965352449
-269.947933841099163679808841099163679809
-96.33503061529271206690561529271206690562
821
848.826259707287933094384707287933094385
730.310130675029868578255675029868578256
-53.01244997013142174432497013142174432497
778.697227449223416965352449223416965352
770.374648804062126642771804062126642772

#### Statement 48: ROUND(BDAY, 'month') FROM EMPLOYEE

ROUND(BDAY, 'month')
01-JUN-60
01-FEB-78
01-DEC-45
01-JUL-31
01-JAN-55
01-SEP-52
01-AUG-62
01-NOV-27
01-AUG-58
01-APR-59

#### Statement 49: TRUNC(BDAY, 'month') FROM EMPLOYEE

TRUNC(BDAY, 'month')
01-JUN-60
01-FEB-78
01-DEC-45
01-JUN-31
01-JAN-55
01-SEP-52
01-JUL-62
01-NOV-27
01-JUL-58
01-MAR-59

#### Statement 50: NEXT\_DAY(BDAY, 'monday') FROM EMPLOYEE

NEXT_DAY(BDAY, MONDAY)
13-JUN-60
13-FEB-78
11-DEC-45
23-JUN-31
10-JAN-55
22-SEP-52
06-AUG-62
15-NOV-27
21-JUL-58
30-MAR-59

#### Statement 51: LAST\_DAY(BDAY) FROM EMPLOYEE

LAST_DAY(BDAY)
30-JUN-60
28-FEB-78
31-DEC-45
30-JUN-31
31-JAN-55
30-SEP-52
31-JUL-62
30-NOV-27
31-JUL-58
31-MAR-59

The screenshot shows the Live SQL interface with a query and its result. The query is `SELECT SYSDATE FROM DUAL`. The result is a single row with the value `09-JUN-23`.

SYSDATE
09-JUN-23

#### 4. Apply SQL queries to retrieve data by using all conversion functions.

The screenshot shows the Live SQL interface with a query and its result. The query is `SELECT TO_CHAR(BDAY, 'DDTH "OF" FMONTH YYYY') FROM EMPLOYEE`. The result is a list of 10 rows showing dates in a specific format.

TO_CHAR(BDAY, 'DDTH "OF" FMONTH YYYY')
09TH OF JUNE 1960
07TH OF FEBRUARY 1978
08TH OF DECEMBER 2045
20TH OF JUNE 2031
09TH OF JANUARY 1955
15TH OF SEPTEMBER 1952
31ST OF JULY 1962
10TH OF NOVEMBER 2027
19TH OF JULY 1958
29TH OF MARCH 1959

The screenshot shows the Live SQL interface with a query and its result. The query is `SELECT TO_NUMBER('123') FROM DEPARTMENT`. The result is a list of 5 rows, all containing the value `123`.

TO_NUMBER('123')
123
123
123
123
123

The screenshot shows the Live SQL interface with a query and its result. The query is `SELECT TO_DATE('SEPTEMBER 9 2009', 'MONTH-DD-YYYY') FROM DUAL`. The result is a single row with the value `09-SEP-09`.

TO_DATE('SEPTEMBER92009', 'MONTH-DD-YYYY')
09-SEP-09

**Aim:** To understand different operators and group functions in SQL

- ```
-- arithmetic operators
```

```
-- set operators
```

The screenshot shows the Live SQL interface. The top navigation bar includes a hamburger menu, the Live SQL logo, and user information (aathira.s2021@vitstudent.ac.in). The main area displays a SQL statement: `SELECT DEPTNUM FROM PROJECT UNION ALL SELECT NUM FROM DEPARTMENT`. Below the statement, the results are shown in a table with the header **DEPTNUM** and 10 rows of data: 1, 3, 5, 4, 5, 1, 2, 3, 4, 5. A 'Download CSV' button is at the bottom left of the results table.

The screenshot shows the Live SQL interface with two statements. Statement 42 is a query that selects DEPTNUM from PROJECT intersected with NUM from DEPARTMENT, resulting in 4 rows. Statement 43 is a query that selects NUM from DEPARTMENT minus DEPTNUM from PROJECT, resulting in 2 rows.

Statement 42: `SELECT DEPTNUM FROM PROJECT INTERSECT SELECT NUM FROM DEPARTMENT`

| DEPTNUM |
|---------|
| 1       |
| 3       |
| 4       |
| 5       |

4 rows selected.

Statement 43: `SELECT NUM FROM DEPARTMENT MINUS SELECT DEPTNUM FROM PROJECT`

| NUM |
|-----|
| 2   |

Download CSV

-- logical operators

The screenshot shows a SQL statement using the BETWEEN logical operator to filter employees by salary. The result shows 3 rows.

Statement 34: `SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY BETWEEN 30000 AND 40000`

| FNAME   | LNAME   |
|---------|---------|
| FRANKIN | WONG    |
| JOHN    | SMITH   |
| RAMESH  | NARAYAN |

Download CSV

3 rows selected.

The screenshot shows a SQL statement using the NOT BETWEEN logical operator to filter employees by salary. The result shows 7 rows.

Statement 35: `SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY NOT BETWEEN 30000 AND 40000`

| FNAME    | LNAME   |
|----------|---------|
| DOUG     | GILBERT |
| JOYCE    | PAN     |
| JENNIFER | WALLACE |
| JOYCE    | ENGLISH |
| JAMES    | BORG    |
| ALICIA   | ZELAYA  |
| AHMAD    | JABBAR  |

Download CSV

7 rows selected.

The screenshot shows a SQL statement using the NOT IN logical operator to filter employees by salary. The result shows 7 rows.

Statement 37: `SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY NOT IN (30000, 40000, 70000)`

| FNAME    | LNAME   |
|----------|---------|
| DOUG     | GILBERT |
| JENNIFER | WALLACE |
| RAMESH   | NARAYAN |
| JOYCE    | ENGLISH |
| JAMES    | BORG    |
| ALICIA   | ZELAYA  |
| AHMAD    | JABBAR  |

Download CSV

7 rows selected.



## -- Comparative operator

Live SQL

My Scripts \ Script

Statement 28

`SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY = 38000`

| FNAME  | LNAME   |
|--------|---------|
| RAMESH | NARAYAN |

Download CSV

Live SQL

My Scripts \ Script

Statement 29

`SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY != 38000`

| FNAME    | LNAME   |
|----------|---------|
| DOUG     | GILBERT |
| JOYCE    | PAN     |
| FRANKIN  | WONG    |
| JENNIFER | WALLACE |
| JOHN     | SMITH   |
| JOYCE    | ENGLISH |
| JAMES    | BORG    |
| ALICIA   | ZELAYA  |
| AHMAD    | JABBAR  |

Download CSV

9 rows selected.

Live SQL

My Scripts \ Script

Statement 30

`SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY < 38000`

| FNAME  | LNAME   |
|--------|---------|
| JOHN   | SMITH   |
| JOYCE  | ENGLISH |
| ALICIA | ZELAYA  |
| AHMAD  | JABBAR  |

Download CSV

4 rows selected.

Live SQL

My Scripts \ Script

Statement 31

`SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY <= 38000`

| FNAME  | LNAME   |
|--------|---------|
| JOHN   | SMITH   |
| RAMESH | NARAYAN |
| JOYCE  | ENGLISH |
| ALICIA | ZELAYA  |
| AHMAD  | JABBAR  |

Download CSV

5 rows selected.

Live SQL

My Scripts \ Script

Statement 32

`SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY > 38000`

| FNAME    | LNAME   |
|----------|---------|
| DOUG     | GILBERT |
| JOYCE    | PAN     |
| FRANKIN  | WONG    |
| JENNIFER | WALLACE |
| JAMES    | BORG    |

Download CSV

5 rows selected.

Live SQL

My Scripts \ Script

Statement 33

`SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY >= 38000`

| FNAME    | LNAME   |
|----------|---------|
| DOUG     | GILBERT |
| JOYCE    | PAN     |
| FRANKIN  | WONG    |
| JENNIFER | WALLACE |
| RAMESH   | NARAYAN |
| JAMES    | BORG    |

Download CSV

6 rows selected.

Live SQL

Feedback Help aathira.s2021@vitstudent.ac.in

My Scripts \ Script

Statement 38

SELECT FNAME, LNAME FROM EMPLOYEE WHERE ADDRESS LIKE 'SUJ'

| FNAME | LNAME   |
|-------|---------|
| DOUG  | GILBERT |
| JOYCE | PAN     |

Download CSV

2 rows selected.

Live SQL

Feedback Help aathira.s2021@vitstudent.ac.in

My Scripts \ Script

Statement 39

SELECT FNAME, LNAME FROM EMPLOYEE WHERE ADDRESS NOT LIKE 'SUJ'

| FNAME    | LNAME   |
|----------|---------|
| FRANKIN  | WONG    |
| JENNIFER | WALLACE |
| JOHN     | SMITH   |
| RAMESH   | NARAIAN |
| JOYCE    | ENGLISH |
| JAMES    | BORG    |
| AUCIA    | ZELAYA  |
| AHMAD    | JABBAR  |

Download CSV

8 rows selected.

## 2. Apply SQL queries to retrieve data by using all aggregate functions.

The screenshot displays the Live SQL web application interface, showing four separate SQL queries and their results. Each query is executed against an 'EMPLOYEE' table, grouped by 'DEPTNUM'.

**Statement 47: MIN(SALARY)**

```
SELECT DEPTNUM, MIN(SALARY)
FROM EMPLOYEE
GROUP BY DEPTNUM
```

| DEPTNUM | MIN(SALARY) |
|---------|-------------|
| 1       | 55000       |
| 2       | 70000       |
| 5       | 25000       |
| 4       | 25000       |
| 3       | 80000       |

5 rows selected.

**Statement 48: MAX(SALARY)**

```
SELECT DEPTNUM, MAX(SALARY)
FROM EMPLOYEE
GROUP BY DEPTNUM
```

| DEPTNUM | MAX(SALARY) |
|---------|-------------|
| 1       | 55000       |
| 2       | 70000       |
| 5       | 40000       |
| 4       | 43000       |
| 3       | 80000       |

5 rows selected.

**Statement 49: AVG(SALARY)**

```
SELECT DEPTNUM, AVG(SALARY)
FROM EMPLOYEE
GROUP BY DEPTNUM
```

| DEPTNUM | AVG(SALARY) |
|---------|-------------|
| 1       | 55000       |
| 2       | 70000       |
| 5       | 33250       |
| 4       | 31000       |
| 3       | 80000       |

5 rows selected.

**Statement 50: SUM(SALARY)**

```
SELECT DEPTNUM, SUM(SALARY)
FROM EMPLOYEE
GROUP BY DEPTNUM
```

| DEPTNUM | SUM(SALARY) |
|---------|-------------|
| 1       | 55000       |
| 2       | 70000       |
| 5       | 133000      |
| 4       | 93000       |
| 3       | 80000       |

5 rows selected.

3. Apply SQL queries to retrieve data by using all aggregate functions and applying a condition over the aggregate function.

The image displays three screenshots of the Live SQL web application interface, showing SQL queries and their results.

**Top Left Screenshot (Statement 52):**

```
SELECT DEPTNUM, MAX(SALARY)
FROM EMPLOYEE
GROUP BY DEPTNUM
HAVING COUNT(*) < 3
```

| DEPTNUM | MAX(SALARY) |
|---------|-------------|
| 1       | 55000       |
| 2       | 70000       |
| 3       | 80000       |

Download CSV  
3 rows selected.

**Top Right Screenshot (Statement 53):**

```
SELECT DEPTNUM, AVG(SALARY)
FROM EMPLOYEE
GROUP BY DEPTNUM
HAVING AVG(SALARY) BETWEEN 30000 AND 50000
```

| DEPTNUM | AVG(SALARY) |
|---------|-------------|
| 5       | 33250       |
| 4       | 31000       |

Download CSV  
2 rows selected.

**Bottom Screenshot (Statement 54):**

```
SELECT DEPTNUM, SUM(SALARY)
FROM EMPLOYEE
GROUP BY DEPTNUM
HAVING SUM(SALARY) > 100000
```

| DEPTNUM | SUM(SALARY) |
|---------|-------------|
| 1       | 55000       |
| 2       | 70000       |
| 5       | 133000      |
| 4       | 93000       |
| 3       | 80000       |

Download CSV  
5 rows selected.