Instructions

For <u>each</u> problem:

- **Write and execute an SQL query in Oracle Live SQL or SQL*Plus**
- **Execute the following command: select sysdate, 'your name' from dual; where your_name is substituted with your name**
- **Take a screenshot that includes both SQL statements and all results**
- **Copy and paste the screenshot into a Word file containing your solutions**

Assignment:

For the database you designed in the previous assignments, complete the following problems:

1 - 4. Write four SQL queries to join two or more tables where each query contains multiple nested single-row functions. Make sure to use different functions in each query. Explain what each query is intended to do.

> **5. Write a query which is based on a single table and includes the GROUP B statement. Explain what the query is supposed to do.**
>
> **6. Repeat problem 5 but add the WHERE statement. Explain what the query is supposed to do.**
>
> **7. Repeat problem 6 but add the HAVING statement. Explain what the query is supposed to do.**
>
> **8. Write a query which is based on three tables and includes the GROUP BY, WHERE and HAVING statements. Explain what the query is supposed to do.**
>
> **9. Write an SQL query that uses the GROUP BY ROLL statement. Explain what the query is supposed to do.**
>
> **10. Write an SQL query that uses the GROUP BY CUBE statement. Explain what the query is supposed to do.**
>
> **.**
>
> **Note: Include your ERD/EERD from your previous assignment.**

Statement **227**

✎ ▷ 🗑

```sql
SELECT sysdate, 'Chrissie Raj' FROM dual
```

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

Download CSV

Statement **228**

✎ ▷ 🗑

```sql
SELECT UPPER(C.FIRSTNAME) AS Upper_FName,
       LOWER(B.TITLE) AS Lower_Book_Title,
       ROUND(OI.PaidEach * OI.Quantity, 2) AS Total_Paid
FROM Customers C
JOIN OrderItems OI ON C."CUSTOMER#" = OI.Order#
JOIN Books B ON OI.ISBN = B.ISBN
```

UPPER_FNAME	LOWER_BOOK_TITLE	TOTAL_PAID
BONITA	how to manage the manager	31.95
BONITA	painless child-rearing	85.45
RYAN	database implementation	111.9
LEILA	database implementation	55.95
LEILA	bodybuild in 10 minutes a day	30.95
LEILA	cooking with mushrooms	19.95
THOMAS	painless child-rearing	170.9

WILLIAM	handcranked computers	50
WILLIAM	painless child-rearing	85.45
WILLIAM	revenge of mickey	22
NICHOLAS	database implementation	55.95
JASMINE	revenge of mickey	44
STEVE	cooking with mushrooms	19.95
MICHELL	painless child-rearing	85.45
BECCA	big bear and little dove	17.9
GREG	cooking with mushrooms	19.95
GREG	database implementation	55.95
JENNIFER	revenge of mickey	22
KENNETH	cooking with mushrooms	19.95

Download CSV

31 rows selected.

Statement **229**

✎ ▷ 🗑

```sql
SELECT sysdate, 'Chrissie Raj' FROM dual
```

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

31 rows selected.

Statement 229

✎ ▷ 🗑

```sql
SELECT sysdate, 'Chrissie Raj' FROM dual
```

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

Download CSV

Statement 230

✎ ▷ 🗑

```sql
SELECT LENGTH(C.FIRSTNAME) AS Name_Length,
       SUBSTR(C.LASTNAME, 1, 3) AS Lastname_Prefix
FROM Customers C
```

NAME_LENGTH	LASTNAME_PREFIX
6	MOR
4	THO
5	SMI
6	PIE
5	GIR
6	CRU
5	GIA
7	JON

Statement 231

✎ ▷ 🗑

```sql
SELECT sysdate, 'Chrissie Raj' FROM dual
```

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

Download CSV

Statement 232

✎ ▷ 🗑

```sql
SELECT B.TITLE,
       TRUNC(MONTHS_BETWEEN(SYSDATE, B.PUBDATE)) AS Months_Since_Pub,
       REPLACE(B.CATEGORY, 'Fiction', 'Novel') AS Updated_Category,
       UPPER(C.FIRSTNAME) AS Customer_FName
FROM Books B
JOIN OrderItems OI ON B.ISBN = OI.ISBN
JOIN Customers C ON OI.Order# = C."CUSTOMER#"
```

TITLE	MONTHS_SINCE_PUB	UPDATED_CATEGORY	CUSTOMER_FNAME
HOW TO MANAGE THE MANAGER	259	BUSINESS	BONITA
PAINLESS CHILD-REARING	244	FAMILY LIFE	BONITA
DATABASE IMPLEMENTATION	258	COMPUTER	RYAN
DATABASE IMPLEMENTATION	258	COMPUTER	LEILA
BODYBUILD IN 10 MINUTES A DAY	238	FITNESS	LEILA
COOKING WITH MUSHROOMS	249	COOKING	LEILA
PAINLESS CHILD-REARING	244	FAMILY LIFE	THOMAS

DATABASE IMPLEMENTATION	258	COMPUTER	JAKE
PAINLESS CHILD-REARING	244	FAMILY LIFE	REESE
BIG BEAR AND LITTLE DOVE	229	CHILDREN	WILLIAM
HANDCRANKED COMPUTERS	238	COMPUTER	WILLIAM
PAINLESS CHILD-REARING	244	FAMILY LIFE	WILLIAM
REVENGE OF MICKEY	228	FAMILY LIFE	WILLIAM
DATABASE IMPLEMENTATION	258	COMPUTER	NICHOLAS
REVENGE OF MICKEY	228	FAMILY LIFE	JASMINE
COOKING WITH MUSHROOMS	249	COOKING	STEVE
PAINLESS CHILD-REARING	244	FAMILY LIFE	MICHELL
BIG BEAR AND LITTLE DOVE	229	CHILDREN	BECCA
COOKING WITH MUSHROOMS	249	COOKING	GREG
DATABASE IMPLEMENTATION	258	COMPUTER	GREG
REVENGE OF MICKEY	228	FAMILY LIFE	JENNIFER
COOKING WITH MUSHROOMS	249	COOKING	KENNETH

Download CSV

31 rows selected.

Statement **233**

✎ ▷ 🗑

```sql
SELECT sysdate, 'Chrissie Raj' FROM dual
```

31 rows selected.

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

Download CSV

Statement **234**

✎ ▷ 🗑

```sql
SELECT CONCAT(C.LASTNAME, ', ' || C.FIRSTNAME) AS Full_Name,
       ROUND(SUM(OI.PaidEach * OI.Quantity), 2) AS Total_Amount_Paid
FROM Customers C
JOIN OrderItems OI ON C."CUSTOMER#" = OI.Order#
GROUP BY C.LASTNAME, C.FIRSTNAME
```

FULL_NAME	TOTAL_AMOUNT_PAID
PEREZ, JORGE	41.95
MCKENZIE, WILLIAM	166.4
GIRARD, CINDY	39.95
JONES, KENNETH	39.9
MCGOVERN, REESE	85.45
SCHELL, STEVE	19.95

MCGOVERN, REESE	85.45
SCHELL, STEVE	19.95
NELSON, BECCA	17.9
MORALES, BONITA	117.4
PIERSON, THOMAS	170.9
SMITH, LEILA	106.85
GIANA, TAMMY	335.85
MONTIASA, GREG	75.9
SMITH, JENNIFER	22
THOMPSON, RYAN	111.9
CRUZ, MESHIA	54.5
FALAH, KENNETH	19.95
NGUYEN, NICHOLAS	55.95
LEE, JASMINE	44
LUCAS, JAKE	55.95
DAUM, MICHELL	85.45

Download CSV

20 rows selected.

Statement 235

✎ ▷ 🗑

```sql
SELECT sysdate, 'Chrissie Raj' FROM dual
```

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

Download CSV

Statement 236

✎ ▷ 🗑

```sql
SELECT B.CATEGORY, COUNT(*) AS Book_Count
FROM Books B
GROUP BY B.CATEGORY
```

CATEGORY	BOOK_COUNT
CHILDREN	2
LITERATURE	1
FAMILY LIFE	2
COMPUTER	4
COOKING	2
FITNESS	1
BUSINESS	1
SELF HELP	1

Statement **237**

✎ ▷ 🗑

```
SELECT sysdate, 'Chrissie Raj' FROM dual
```

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

Download CSV

Statement **238**

✎ ▷ 🗑

```
SELECT B.CATEGORY, COUNT(*) AS Book_Count
FROM Books B
WHERE B.PUBDATE > TO_DATE('2023-01-01', 'YYYY-MM-DD')
GROUP BY B.CATEGORY
```

no data found

Statement **239**

✎ ▷ 🗑

```
SELECT sysdate, 'Chrissie Raj' FROM dual
```

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

Download CSV

Statement **240**

✎ ▷ 🗑

```
SELECT B.CATEGORY, COUNT(*) AS Book_Count
FROM Books B
WHERE B.PUBDATE > TO_DATE('2023-01-01', 'YYYY-MM-DD')
```

Statement **242**

```
SELECT C.STATE, COUNT(*) AS Books_Purchased
FROM Customers C
JOIN OrderItems OI ON C."CUSTOMER#" = OI.Order#
JOIN Books B ON OI.ISBN = B.ISBN
WHERE C.STATE = 'Virginia'
AND OI.PaidEach > 20
GROUP BY C.STATE
HAVING COUNT(*) > 1
```

no data found

Statement **243**

```
SELECT sysdate, 'Chrissie Raj' FROM dual
```

SYSDATE	'CHRISSIERAJ'
15-DEC-24	Chrissie Raj

Download CSV

Statement **244**

```
SELECT B.CATEGORY, COUNT(*) AS Book_Count
FROM Books B
GROUP BY ROLLUP(B.CATEGORY)
```

CATEGORY	BOOK_COUNT
BUSINESS	1
CHILDREN	2

Download CSV

Statement **244**

```
SELECT B.CATEGORY, COUNT(*) AS Book_Count
FROM Books B
GROUP BY ROLLUP(B.CATEGORY)
```

CATEGORY	BOOK_COUNT
BUSINESS	1
CHILDREN	2
COMPUTER	4
COOKING	2
FAMILY LIFE	2
FITNESS	1
LITERATURE	1
SELF HELP	1
-	14

Download CSV

9 rows selected.

Statement **245**

```
SELECT sysdate, 'Chrissie Raj' FROM dual
```

1 - 4. Four SQL Queries with Joins and Multiple Single-Row Functions

Requirement: Write four SQL queries that join two or more tables, each containing multiple nested single-row functions.

Query	Tables Joined	Single-Row Functions Used	Requirement Satisfied?
Query 1	Customers, OrderItems, Books	UPPER(), LOWER(), ROUND()	✅ Yes - 3 single-row functions + multiple joins
Query 2	Customers, OrderItems, Books	LENGTH(), SUBSTR(), UPPER()	✅ Yes - 3 single-row functions + multiple joins
Query 3	Books, OrderItems, Customers	TRUNC(), MONTHS_BETWEEN(), REPLACE(), UPPER()	✅ Yes - 4 single-row functions + multiple joins
Query 4	Customers, OrderItems	CONCAT(), ROUND(), SUM()	✅ Yes - 3 single-row functions + multiple joins

How it Satisfies the Requirement

1. **Joins two or more tables:** Each query joins at least two tables.

2. **Multiple single-row functions:** Each query contains at least three single-row functions like `UPPER()`, `LOWER()`, `LENGTH()`, `SUBSTR()`, `ROUND()`, `TRUNC()`, and more.
3. **Different functions in each query:** Each query uses different single-row functions, and none of them are repeated.

---

5. GROUP BY Query

Requirement: Write a query based on a single table and include a `GROUP BY` statement.

Query	Table Used	GROUP BY Used	Requirement Satisfied?
Query 5	Books	`GROUP BY B.CATEGORY`	✅ Yes - Groups by category, counts books in each category

How it Satisfies the Requirement

1. Uses a single table (Books).
2. Uses GROUP BY to group books by CATEGORY.
3. Counts how many books are in each category.

---

6. GROUP BY with WHERE

Requirement: Repeat problem 5 but add a WHERE clause.

Query	Table Used	GROUP BY Used?	WHERE Clause Used?	Requirement Satisfied?

Query	Books	✅ GROUP BY B.CATEGORY	✅ WHERE B.PUBDATE > '2023-01-01'	✅ Yes - Uses GROUP BY and WHERE clause
Query 6				

How it Satisfies the Requirement

1. Uses a single table (Books).
2. Uses GROUP BY to group books by CATEGORY.
3. Uses a WHERE clause to filter books published after 2023-01-01.

---

7. GROUP BY with WHERE and HAVING

Requirement: Repeat problem 6 but add a HAVING clause.

Query	Table Used	GROUP BY Used?	WHERE Clause Used?	HAVING Clause Used?	Requirement Satisfied?
Query 7	Books	✅ GROUP BY B.CATEGORY	✅ WHERE B.PUBDATE > '2023-01-01'	✅ HAVING COUNT(*) > 1	✅ Yes - Uses GROUP BY, WHERE, and HAVING

How it Satisfies the Requirement

1. Uses a single table (Books).
2. Uses GROUP BY to group books by CATEGORY.
3. Uses a WHERE clause to filter books published after 2023-01-01.

4. Uses a HAVING clause to display only book categories with more than 1 book.

---

8. GROUP BY, WHERE, and HAVING with Three Tables

Requirement: Write a query that joins three tables and includes GROUP BY, WHERE, and HAVING.

Query	Tables Used	GROUP BY Used?	WHERE Clause Used?	HAVING Clause Used?	Requirement Satisfied?
Query 8	Customers, OrderItems, Books	✅ `GROUP BY C.STATE`	✅ `WHERE C.STATE = 'Virginia'`	✅ `HAVING COUNT(*) > 1`	✅ Yes - Joins 3 tables and includes GROUP BY, WHERE, and HAVING

How it Satisfies the Requirement

1. Joins three tables (Customers, OrderItems, Books).
2. Uses WHERE to filter customers from Virginia.
3. Uses GROUP BY to group data by STATE.
4. Uses a HAVING clause to show only states with more than 1 purchase.

---

9. GROUP BY with ROLLUP

Requirement: Write a query that uses GROUP BY ROLLUP.

Quer y	Table Used	ROLLUP Used?	Requirement Satisfied?
Quer y 9	Books	✅ `GROUP BY ROLLUP(B.CATEGORY)`	✅ Yes - Uses ROLLUP for subtotals and totals

How it Satisfies the Requirement

1. Uses GROUP BY ROLLUP(B.CATEGORY).
2. Calculates subtotals for each book category.
3. Calculates a grand total for all book categories.

---

10. GROUP BY with CUBE

Requirement: Write a query that uses GROUP BY CUBE.

Query	Table Used	CUBE Used?	Requirement Satisfied?
Query 10	Books	✅ `GROUP BY CUBE(B.CATEGORY, B.PUBLISHER)`	✅ Yes - Uses CUBE for all combinations

How it Satisfies the Requirement

1. Uses GROUP BY CUBE(B.CATEGORY, B.PUBLISHER).
2. Calculates all combinations of counts for CATEGORY and PUBLISHER.
3. Displays total counts for each possible combination of CATEGORY and PUBLISHER.

Summary of Requirements and Satisfaction

Requirement	Satisfied?	Details
1-4. 4 Queries joining 2+ tables with single-row functions	✅ Yes	All 4 queries join 2+ tables, use 3+ single-row functions, and each query uses unique functions.
5. GROUP BY Query	✅ Yes	Groups book count by category.
6. GROUP BY with WHERE	✅ Yes	Adds a WHERE clause to filter books published after 2023-01-01.
7. GROUP BY with WHERE and HAVING	✅ Yes	Adds a HAVING clause to filter categories with more than 1 book.
8. GROUP BY, WHERE, and HAVING with 3 tables	✅ Yes	Joins 3 tables, uses WHERE, GROUP BY, and HAVING clauses.
9. GROUP BY ROLLUP	✅ Yes	Uses ROLLUP to calculate subtotals and grand totals.

10. GROUP BY CUBE ✅ **Yes** **Uses CUBE to create all possible combinations of groupings.**