

# SQL Practices

## 1. List Department details (ID, Name, Location) which does not have any employees

**Solution:** Use xx1553\_departments and xx1553\_employees. Get the department\_id which are not in employees table. It results the departments which doesn't have employees.

**Query:** SELECT

department\_id,

department\_name,

location\_id

FROM

xx1553\_departments

WHERE

department\_id NOT IN (

SELECT

department\_id

FROM

xx1553\_employees);

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	6	Admin	1
2	7	Cybersecurity	2
3	5	Finance	2

## 2. List all employees whose salary is greater than average salary of all employees

**Solution:** Use xx1553\_employees table. First write sub-query for average salary of all employees and write the outer query to get salary of each employees and apply the condition where salary of each employees greater than the inner sub-query.

**Query:** SELECT

first\_name || last\_name AS emp\_name,

```

salary
FROM
  xx1553_employees
WHERE
  salary > (
    SELECT
      AVG(salary) FROM xx1553_employee);

```

	EMP_NAME	SALARY
1	EmilyWilliams	70000
2	Rock	70000
3	RohitSharma	75000
4	ViratKohli	80000
5	MitchellMarsh	80000

### 3. List all employees who are getting the lowest salary

**Solution:** Use xx1553\_employees table. First write sub-query to lowest salary and write the outer query to get salary of each employees where salary is equal to inner sub-query.

**Query:** SELECT

```

  first_name || last_name AS emp_name,
  salary
FROM
  xx1553_employees
WHERE
  salary = (
    SELECT
      MIN(salary)
    FROM

```

xx1553\_employees

);

	EMP_NAME	SALARY
1	ChrisBrown	50000
2	JohnCena	50000

#### 4. List customer wise sales

**Solution:** Here we should take two tables, xx1553\_customers & xx1553\_sales. Join two tables by equi join using common column in both tables called Customer\_id. Perform Group by function using Customer\_name and perform aggregation function(sum) on sale\_amount.

**Query:** SELECT

c.customer\_name,

SUM(s.sale\_amount) sales

FROM

xx1553\_customers c,

xx1553\_sales s

WHERE

c.customer\_id = s.customer\_id

GROUP BY

c.customer\_name;

	CUSTOMER_NAME	SALES
1	Suresh Raina	202396
2	Ravindra Jadeja	145000
3	Rishabh Pant	123620
4	Rahul Dravid	111957
5	Ruturaj Gaikwad	107850
6	Shubman Gill	82400
7	MS Dhoni	50562
8	Virat Kohli	18500

### 5. List Year wise, month wise Sales

**Solution:** Use xx1553\_sales table. Extract year from sale\_date. Perform Group by function on year and aggregation (sum) on sale\_amount column. Hence get the required year wise sales.

**Query:** SELECT

to\_char(sale\_date, 'YYYY') AS year,

SUM(sale\_amount) AS sales

FROM

xx1553\_sales

GROUP BY

to\_char(sale\_date, 'YYYY');

	YEAR	SALES
1	2023	330632
2	2021	167795
3	2020	160896
4	2019	136162
5	2022	46800

**Solution:** Use xx1553\_sales table. Extract month from sale\_date. Perform Group by function on month and aggregation (sum) on sale\_amount column. Hence get the required month wise sales.

**Query:** SELECT

to\_char(sale\_date, 'MON') AS month,

SUM(sale\_amount) AS sales

FROM

xx1553\_sales

GROUP BY

to\_char(sale\_date, 'MON');

	MONTH	SALES
1	NOV	261021
2	JUN	186194
3	SEP	163600
4	JUL	111470
5	FEB	70000
6	OCT	50000

#### 6. List Year wise, month wise Direct Sales, Online Sales separately

**Solution:** Use xx1553\_sales table. Extract year from sale\_date. Perform Group by function on year and aggregation (count \*). In where condition give sale\_mode = 'Direct Sales' . Hence get the required year wise Direct Sales count.

**Query:** SELECT

to\_char(sale\_date, 'YYYY') AS year,  
COUNT(\*) AS sales\_count

FROM

xx1553\_sales

WHERE

sale\_mode = 'Direct Sales'

GROUP BY

to\_char(sale\_date, 'YYYY');

	YEAR	SALES_COUNT
1	2023	3
2	2019	2
3	2021	1
4	2022	1
5	2020	1

**Solution:** Use xx1553\_sales table. Extract year from sale\_date. Perform Group by function on year and aggregation (count \*). In where condition give sale\_mode = 'Online Sales' . Hence get the required year wise Online Sales count.

**Query:** SELECT

```
to_char(sale_date, 'YYYY') AS year,  
COUNT(*)           AS sales_count
```

FROM

```
xx1553_sales
```

WHERE

```
sale_mode = 'Online Sales'
```

GROUP BY

```
to_char(sale_date, 'YYYY');
```

	YEAR	SALES_COUNT
1	2020	3
2	2021	2
3	2023	2
4	2019	1

**Solution:** Use xx1553\_sales table. Extract month from sale\_date. Perform Group by function on month and aggregation (count \*). In where condition give sale\_mode = 'Direct Sales' . Hence get the required month wise Direct Sales count.

**Query:** SELECT

```
to_char(sale_date, 'MON') AS month,  
COUNT(*)           AS sales_count
```

FROM

```
xx1553_sales
```

WHERE

```
sale_mode = 'Direct Sales'
```

GROUP BY

to\_char(sale\_date, 'MON');

	MONTH	SALES_COUNT
1	NOV	3
2	JUN	2
3	JUL	1
4	SEP	1
5	OCT	1

**Solution:** Use xx1553\_sales table. Extract month from sale\_date. Perform Group by function on month and aggregation (count \*). In where condition give sale\_mode = 'Online Sales' . Hence get the required month wise Online Sales count.

**Query:** SELECT

to\_char(sale\_date, 'MON') AS month,

COUNT(\*) AS sales\_count

FROM

xx1553\_sales

WHERE

sale\_mode = 'Online Sales'

GROUP BY

to\_char(sale\_date, 'MON');

	MONTH	SALES_COUNT
1	NOV	3
2	SEP	2
3	JUL	1
4	JUN	1
5	FEB	1

### 7. List customers who are exceeding their credit limits

**Solution:** Use xx1553\_customers table. Check current\_balance with credit\_limit using where condition. If current\_balance is greater than credit\_limit, print respective customer\_name.

**Query:** SELECT

customer\_name

FROM

xx1553\_customers

WHERE

current\_balance > credit\_limit;

	CUSTOMER_NAME
1	Rahul Dravid
2	MS Dhoni
3	Virat Kohli
4	Ravindra Jadeja
5	Ravichandran Ashwin

### 8. List all employees who were holding more than one Job in various periods in the company

**Solution:** Use xx1553\_job\_history table and xx1553\_employees table. Join two tables by equi join using same column in both tables called employee\_id.

Perform Group by in employee\_id, company\_id, company\_name from xx1553\_job\_history table and first\_name and last\_name from xx1553\_employees table.

Perform having function and give condition count(\*) greater than 1.

In select statement, give employee\_id, company\_id, company\_name from xx1553\_job\_history table and concatenating first\_name and last\_name from xx1553\_employees table and perform aggregation function (count (\*)).

Hence, we get employees who were holding more than one Job in various periods in the company.

**Query:** SELECT

jh.employee\_id,

e.first\_name || e.last\_name AS name,

jh.company\_id,

jh.company\_name,



```

COUNT(*) employed_times_in_same_company
FROM
    xx1553_job_history jh,
    xx1553_employees e
WHERE
    e.employee_id = jh.employee_id
GROUP BY
    e.first_name || e.last_name,
    jh.employee_id,
    jh.company_name,
    jh.company_id
HAVING
    COUNT(*) > 1;

```

	EMPLOYEE_ID	NAME	COMPANY_ID	COMPANY_NAME	EMPLOYED_TIMES_IN_SAME_COMPANY
1	2	JaneSmith	14i	apps	2
2	1	JohnDoe	14i	apps	2
3	4	EmilyWilliams	3	Infosys	3

### 9. List all employees with their first job

**Solution:** Use xx1553\_job\_history table. By using over partition by, group employee\_id.

Order by employee\_id and start\_date.

Use count as aggregation function. Naming alias as ct to count function. Store this in a with clause as temp.

Write another query using this with clause temp and xx1553\_employees table. Perform inner join to get name of the employees using common column called employee\_id.

After that in where clause give a condition called ct=1.

By then, able to get all employees with their first job.

**Query:** WITH temp AS (  
 SELECT  
 employee\_id,  
 start\_date,  
 company\_name,  
 job\_id,  
 COUNT(\*)

```

OVER(PARTITION BY employee_id
ORDER BY
employee_id, start_date
) AS ct
FROM
xx1553_job_history
)
SELECT
t.employee_id, first_name || ' ' || last_name as employee_name,
t.company_name, start_date, t.job_id
FROM
temp t join xx1553_employees e
on t.employee_id=e.employee_id
WHERE
ct = 1;

```

	EMPLOYEE_ID	EMPLOYEE_NAME	COMPANY_NAME	START_DATE	JOB_ID
1	1	John Doe	4i apps	01-01-22	IT_PROG
2	2	Jane Smith	4i apps	15-03-22	SA_REP
3	3	Michael Johnson	Mu Sigma	01-12-21	HR_REP
4	4	Emily Williams	Infosys	01-01-19	HR
5	5	Chris Brown	TCS	01-03-21	JR_DEV

#### 10. How any “orderable” products available

**Solution:** Use xx1553\_inventories table. Give condition as quantity is not null in where clause.

By then, able to get orderable products.

**Query:** SELECT

product\_id,

quantity

FROM

xx1553\_inventories

WHERE

quantity IS NOT NULL;

	PRODUCT_ID	QUANTITY
1	LE256	2
2	RA652	3
3	PE536	4
4	TT235	7
5	BA536	4

### 11. How to find top three highest salary in emp table in oracle?

**Solution:** Use xx1553\_employees table. Get distinct salary. Because many employees may get different salaries. Order by salary in descending and used fetch condition to fetch first three rows.

**Query:** SELECT DISTINCT

salary

FROM

xx1553\_employees

ORDER BY

salary DESC

FETCH FIRST 3 ROWS ONLY;

	SALARY
1	80000
2	75000
3	70000

### 12. SQL Query to find fifth highest salary with empno

**Solution:** Use xx1553\_employees table. Using dense\_rank() function, rank the employee\_salaries.

Why particularly dense\_rank means...?. Dense\_rank won't skip the rank. If two or more employees have the same salary means, Rank will be same for those employees.

By then, we can be able to get the fifth highest salary with minimum one employee.

Created with clause as cte. By using cte, in where clause give rn=5.

It fetches the fifth highest salary.

**Query:** WITH cte AS (

SELECT

employee\_id,

salary,

DENSE\_RANK()

OVER(

ORDER BY

salary DESC

) rnk

FROM

xx1553\_employees

)

SELECT

\*

FROM

cte

WHERE

rnk = 5;

	EMPLOYEE_ID	SALARY	RNK
1	1	60000	5

### 13. What is the total on-hand quantity of all products

**Solution:** Use xx1553\_inventories table. By aggregation function(sum), we can be able to add total on-hand quantity of all products.

**Query:** SELECT

SUM(quantity) total\_on\_hand\_quantity

FROM

xx1553\_inventories;

	TOTAL_ON_HAND_QUANTITY
1	62

#### 14. List the products does not have stock

**Solution:** Use xx1553\_inventories and xx1553\_products table. xx1553\_products table shows the name of all products.

xx1553\_inventories table provides stock of those products. Join two tables using equi join. Give condition quantity is not null. By then, we can get products that do not have stock.

**Query:** SELECT

product\_name

FROM

xx1553\_inventories i,

xx1553\_products p

WHERE

p.product\_id = i.product\_id

AND i.quantity IS NULL;

	PRODUCT_NAME
1	Full sleeve formal shirt
2	Men Cotton Half Sleeve Slim Fit Polo Neck White Vivid Polo
3	Casual shirt Knick Knack Nook

#### 15. List the items which can be ordered

**Solution:** Use xx1553\_inventories table. In where clause, put quantity is not null. By then, we can be able to list the items which can be ordered.

**Query:** SELECT

product\_id

FROM

xx1553\_inventories

WHERE

quantity IS NOT NULL;

	PRODUCT_ID
1	LE256
2	RA652
3	PE536
4	TI235
5	BA536

#### 16. Get the order details for one order

**Solution:** Use xx1553\_orders table. Randomly select one order (order\_id=2) and retrieve details of that order using where clause.

**Query:** SELECT

\*

FROM

xx1553\_orders

WHERE

order\_id = 2;

ORDER_ID	PRODUCT_ID	PRODUCT_NAME	QUANTITY	UNIT_PRICE	TOTAL_PRICE	PAYMENT_STATUS
1	2 LE256	Men Levis T-Shirts Collar Polo Round Neck	1	1399	1399	Paid

#### 17. Verify whether the order\_total is calculated correctly or not

**Solution:** Use xx1553\_orders table. In this table quantity, unit price, total price are mentioned.

Using case when multiply quantity and unit price column, if result is equal to total price column, then print correct. vice versa.

**Query:** SELECT

quantity,

unit\_price,

total\_price,

CASE

WHEN ( quantity \* unit\_price ) = total\_price THEN

'CORRECT'

ELSE

'INCORRECT'

END calculated\_correctly\_or\_not

FROM

xx1553\_orders;

	QUANTITY	UNIT_PRICE	TOTAL_PRICE	CALCULATED_CORRECTLY_OR_NOT
1	2	1499	1599	INCORRECT
2	1	1399	1399	CORRECT
3	3	1799	4362	INCORRECT
4	1	1100	1100	CORRECT
5	2	799	1598	CORRECT
6	1	599	750	INCORRECT

#### 18. List the items which are ordered

**Solution:** Use xx1553\_products and xx1553\_orders table. Retrieve the product id which is present in xx1553\_orders table using in operator. Then able to get the items which are ordered.

**Query:** SELECT

product\_id,

product\_name

FROM

xx1553\_products

WHERE

product\_id IN (

SELECT

product\_id

FROM

xx1553\_orders

);

	PRODUCT_ID	PRODUCT_NAME
1	OT123	Full sleeve formal shirt
2	LE256	Men Levis T-Shirts Collar Polo Round Neck
3	RA652	Faded Men Jeans Pant Narrow fit
4	PE536	Men multi color slim fit breathable checked full sleeves formal cotton shirt
5	CL563	Men Cotton Half Sleeve Slim Fit Polo Neck White Vivid Polo
6	TT235	Full sleeve blue shirt

### 19. List of items which are not yet ordered

**Solution:** Use xx1553\_products and xx1553\_orders table. Retrieve the product id which is not present in xx1553\_orders table using not in operator. Then able to get the items which are not yet ordered.

**Query:** SELECT

product\_id,

product\_name

FROM

xx1553\_products

WHERE

product\_id NOT IN (

SELECT

product\_id

FROM

xx1553\_orders

);

	PRODUCT_ID	PRODUCT_NAME
1	BA536	Formal full sleeve cotton shirt
2	KKN523	Casual shirt Knick Knack Nook

### 20. List the Order details where items are ordered less than the list price

**Solution:** Use xx1553\_orders and xx1553\_products table. Retrieve the order details where unit price which is mentioned in xx1553\_orders table is less than product price which is mentioned in xx1553\_products table.

**Query:** SELECT



\*

FROM

xx1553\_orders o

WHERE

unit\_price < (

SELECT

product\_price

FROM

xx1553\_products p

WHERE

o.product\_id = p.product\_id

);

ORDER_ID	PRODUCT_ID	PRODUCT_NAME	QUANTITY	UNIT_PRICE	TOTAL_PRICE	PAYMENT_STATUS
1	4FE536	Men multi color slim fit breathable checked full sleeves formal cotton shirt	1	1100	1100	Paid
2	6TT235	Full sleeve blue shirt	1	599	750	Paid

## 21. List the Order details where items are ordered less than the minimum price

**Solution:** Use xx1553\_orders and xx1553\_products table. Retrieve the order details where unit price mentioned in xx1553\_orders table is less than the minimum product price of xx1553\_products table using where clause.

**Query:** SELECT

\*

FROM

xx1553\_orders

WHERE

unit\_price < (

SELECT

MIN(product\_price)

FROM

xx1553\_products

);

ORDER_ID	PRODUCT_ID	PRODUCT_NAME	QUANTITY	UNIT_PRICE	TOTAL_PRICE	PAYMENT_STATUS
1	6	TT235	Full sleeve blue shirt	1	599	750 Paid

## 22. Find the profit of each order line (compare minimum price with order)

**Solution:** Use xx1553\_orders and xx1553\_products table. Join two tables by equi join using common column called product id. Calculate a new column called outcome\_price by subtracting product price of xx1553\_products table from unit price of xx1553\_ordres table.

Create another column called profit\_loss using case when.

In profit\_loss column, print profit if the value of outcome price is greater than 0 and loss if it is less than 0. if it is equal to 0 print No profit/loss.

**Query:** SELECT

p.product\_price,

unit\_price AS sale\_price,

( o.unit\_price - p.product\_price ) AS outcome\_price,

CASE

WHEN ( o.unit\_price - p.product\_price ) > 0 THEN

'profit'

WHEN ( o.unit\_price - p.product\_price ) < 0 THEN

'loss'

ELSE

'No profit/loss'

END profit\_loss

FROM

xx1553\_orders o,

xx1553\_products p

WHERE

o.product\_id = p.product\_id;

	PRODUCT_PRICE	SALE_PRICE	OUTCOME_PRICE	PROFIT_LOSS
1	1499	1499	0	No profit/loss
2	1399	1399	0	No profit/loss
3	1799	1799	0	No profit/loss
4	1199	1100	-99	loss
5	699	799	100	profit
6	699	599	-100	loss

### 23. Find the profit of each order and its %

**Solution:** Use xx1553\_orders and xx1553\_products table. Join two tables by equi join using common column called product id. Calculate a new column called profit\_loss by subtracting product price of xx1553\_products table from unit price of xx1553\_ordres table.

Using case when, if the value of profit\_loss column is greater than 0 then divide the value by product\_price and multiply by 100. Otherwise print 0.

Results the profit percentage.

**Query:** SELECT

product\_price,

unit\_price - product\_price AS profit\_loss,

CASE

WHEN ( unit\_price - product\_price ) > 0 THEN

round(((unit\_price - product\_price) / product\_price) \* 100)

ELSE

0

END profit\_percent

FROM

xx1553\_products p,

xx1553\_orders o

WHERE

p.product\_id = o.product\_id;

	PRODUCT_PRICE	PROFIT_LOSS	PROFIT_PERCENT
1	1499	0	0
2	1399	0	0
3	1799	0	0
4	1199	-99	0
5	699	100	14
6	699	-100	0

#### 24. Create table xx100\_product by copying only orderable items from product master

**Solution:** Create a new table called xx100\_product using create table syntax.

Write query to get orderable items by searching product\_id from xx1553\_orders table in xx1553\_products.

**Query:** CREATE TABLE xx100\_product

AS

(

SELECT

\*

FROM

xx1553\_orders

WHERE

product\_id IN (

SELECT

product\_id

FROM

xx1553\_products

)

);

ORDER_ID	PRODUCT_ID	PRODUCT_NAME	QUANTITY	UNIT PRICE	PRICE	PAYMENT STATUS
1	1 OT123	Full sleeve formal shirt	2	1499	2998	Paid
2	2 LE256	Men Levis T-Shirts Collar Polo Round Neck	1	1399	1399	Paid
3	3 RA652	Faded Men Jeans Pant Narrow fit	3	1799	5397	Pending
4	4 PE536	Men multi color slim fit breathable checked full sleeves formal cotton shirt	1	1100	1100	Paid
5	5 CL563	Men Cotton Half Sleeve Slim Fit Polo Neck White Vivid Polo	2	599	1198	Paid
6	6 TT235	Full sleeve blue shirt	1	620	620	Paid

## 25. Take backup of employee master

**Solution:** To take backup, we can create view as backup\_employees for xx1553\_employees table.

**Query:** CREATE OR REPLACE VIEW backup\_employees AS

SELECT

\*

FROM

xx1553\_employees;

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	MANAGER_ID	DEPARTMENT_NAME	EXPERIENCE
1	John	Doe	john.doe@gmail.com	8536259696	15-05-22	JD123	60000	1	(null)	Sales	5 years
2	Jane	Smith	jane.smith@gmail.com	7586936363	20-03-22	JS456	65000	2	1	Marketing	8 years
3	Michael	Johnson	michael.johnson@yahoo.com	9856365989	10-07-22	MJ789	58000	1	2	Sales	2 years
4	Emily	Williams	emily.williams@gmail.com	8889659959	08-01-22	EW101	70000	3	3	HR	No experience
5	Chris	Brown	chris.brown@yahoo.com	8989565858	25-09-22	CB111	50000	2	4	Marketing	3 years
6	John	Cena	john.cena@gmail.com	6969536952	20-02-22	AD236	50000	4	5	IT	2 years
7	Rock	(null)	rock@gmail.com	8989656325	23-03-23	GH569	70000	2	6	Marketing	No experience
8	Virat	Kohli	virat.kohli@gmail.com	8795362359	11-04-23	RT569	80000	4	7	IT	No experience
9	Rohit	Sharma	rohit.sharma@gmail.com	8795885659	10-02-23	KK856	75000	1	2	Sales	3 years
10	Mitchell	Marsh	mittchell.marsh@gmail.com	9966885955	05-06-23	GT567	80000	3	3	HR	No experience

## 26. Create table xx100\_employee with (id, full\_name, salary) and copy data from employee master

**Solution:** Create a table called xx100\_employee with columns called id, full\_name, salary with create table syntax from xx1553\_employees table.

**Query:** CREATE TABLE xx100\_employee

AS

(

SELECT

employee\_id AS id,

first\_name || last\_name AS full\_name,

salary

FROM

xx1553\_employees

);

	ID	FULL_NAME	SALARY
1	1	JohnDoe	60000
2	2	JaneSmith	65000
3	3	MichaelJohnson	58000
4	4	EmilyWilliams	70000
5	5	ChrisBrown	50000
6	6	JohnCena	50000
7	7	Rock	70000
8	8	ViratKohli	80000
9	9	RohitSharma	75000
10	10	MitchellMarsh	80000

**27. In new table xx100\_employee increment salary by 10%**

**Solution:** Create a new table called xx100\_employee.

Create a new column called salary\_increment\_by\_10\_percent in select statement by calculating 10% of salary and adding the value to salary.

Results in salary increment by 10 percent.

**Query:** SELECT

salary,

salary \* ( 10 / 100 ) AS ten\_percentage,

salary + ( salary \* ( 10 / 100 ) ) AS salary\_increment\_by\_10\_percent

FROM

xx100\_employee;

	⌄ SALARY	⌄ TEN_PERCENTAGE	⌄ SALARY_INCREMENT_BY_10_PERCENT
1	60000	6000	66000
2	65000	6500	71500
3	58000	5800	63800
4	70000	7000	77000
5	50000	5000	55000
6	50000	5000	55000
7	70000	7000	77000
8	80000	8000	88000
9	75000	7500	82500
10	80000	8000	88000