Exercise 1 : SQL Fundamentals

Database : employees_db

1. SELECT *
FROM employees;

2. SELECT DISTINCT department
FROM employees;

Department
IT
HR
Finance
Marketing

3. SELECT first_name,
last_name,
salary
FROM employees
ORDER BY 'salary' DESC;

first_name	last_name	salary
Emily	Davis	62 000
Mike	Johnson	60 000
Robert	Wilson	59 000
John	Doe	55 000
Sarah	Brown	53 000
Daniel	Clark	53 000
David	White	52 000

4. SELECT first_name,
last_name,
salary
FROM employees
ORDER BY "salary" DESC
LIMIT 5;

5. SELECT first_name,
last_name,
department
FROM employees
WHERE "department" = 'IT';

first_name	last_name	department
John	Doe	IT
Sarah	Brown	IT
Emily	Davis	IT
Laura	Hall	IT

. SELECT first_name,
last_name,
department,
salary
FROM employees
WHERE department = 'finance'
AND salary > 58 000;

f_n	l_n	dept	salary
Mike	Johnson	finance	60 000
Robert	Wilson	finance	59 000

→ first_name | last_name | dept

7. SELECT first_name,
last_name,
department
FROM employees
WHERE department = 'HR' OR 'Marketing';

first_name	last_name	department
Jane	~~Doe~~ Smith	HR
Jessica	Moore	HR
David	White	Marketing
Daniel	Clark	Marketing

8. SELECT first_name,
last_name,
department
FROM employees
WHERE department ~~NOT~~ != 'IT';

first_name	last_name	department
Jane	Smith	HR
Mike	Johnson	Finance
David	White	Marketing
Robert	Wilson	Finance
Jessica	Moore	HR
Daniel	Clark	Marketing

9. SELECT first_name,
last_name,
department
FROM employees
WHERE department IN ('HR', 'IT', 'Finance');

first_name	last_name	department
John	Doe	IT
Jane	Smith	HR
Mike	Johson	Finance
Sarah	Brown	IT
Emily	Davis	IT
Robert	Wilson	finance
Jessica	Moore	HR
Laura	Hall	IT

10. SELECT first_name,
last_name,
department,
salary,
city
FROM employees
WHERE department = 'IT'
AND salary > 50 000
AND city = 'New York';

first_name	last_name	department	salary	city
John	Doe	IT	55 000	New York
Sarah	Brown	IT	53 000	New York

11. SELECT first_name,
last_name,
department,
salary
FROM employees
WHERE department = 'Finance' OR 'Marketing'
AND salary > 52000
ORDER BY salary DESC;

first_name	last_name	department	salary
Mike	Johson	Finance	60 000
Robert	Wilson	Finance	59 000
Daniel	Clark	Maketing	53 000

12. SELECT DISTINT ~~city,~~ department,
~~FROM~~ ~~first_name,~~ city
~~last_name~~
~~department~~
FROM employees
WHERE department IN ('Finance', 'Marketing');

department	city
Finance	Los Angeles
Marketing	San fransisco
Finance	Houston
Marketing	Chicago

13. SELECT first_name,
last_name,
department,
salary,
hire_date,
FROM employees
WHERE department NOT IN ('Finance') AND salary > 50000
ORDER BY hire_date ASC;

first_name	last_name	department	salary	hire_date
Emily	Davis	IT	62 000	2015-02-14
David	White	Marketing	52 000	2016-04-10
Jessica	Moore	HR	51 000	2018-05-22
Sarah	Brown	IT	53 000	2021-03-25
Daniel	Clark	Marketing	53000	2022-06-01

14. SELECT first_name,
last_name,
department,
city
FROM employees
WHERE department IN ('IT', 'Marketing')
AND city IN ('Chicago', 'Los Angeles')
LIMIT 3;

SELECT *
FROM employees
WHERE city IN ('Chicago', 'Los Angeles')
AND department IN ('IT', 'Marketing')
LIMIT 3;

Exercise 2 Aggregate functions

1. SELECT DISTINCT department
FROM students;

department
IT
HR
Finance

2. SELECT ~~AUG (age) AS~~ department,
AUG (Age) AS avg_age
FROM students
GROUP By department;

department	avg_age
IT	20,5
HR	22
Finance	23

3. SELECT department
COUNT (student_id) AS student_count
GROUP BY department
HAVING student_count > 1;

department	student_count
IT	2
HR	2

HAVING statement

↳ it filters data that is aggregated.

employees

Id	Name	department	salary
1	Mark	HR	10
2	Steve	HR	20
3	Jeff	IT	30
4	ELON	IT	25
5	Rothi	France	100

```
SELECT department,
       SUM(salary) AS total_salary
FROM employees
GROUP BY department;
```

department	total_salary
HR	30
IT	55
Finance	100

```
SELECT department,
       SUM(salary) AS total_salary
FROM employees
GROUP BY department
HAVING SUM(salary) > 50;
```

department	total_salary
IT	55
Finance	100

5. SELECT student_id,
name,
age,
department
FROM students
WHERE department IN ('IT', 'HR')
AND Age > 21;

id	name	age	dept

4. SELECT student_id,
name,
age,
department
FROM students
WHERE age BETWEEN 21 AND 23;

6. SELECT department,
SUM(credits) AS total_credits
FROM courses
GROUP BY department
HAVING ~~credits~~ → SUM(credits) > 5;

department	total_credits	
IT	10	✓
~~Finance~~	~~X~~	← below 5
~~HR~~	~~X~~	

7. SELECT course_id,
course_name,
department,
credits
FROM ~~depo coo~~ courses
WHERE credits <>4;

course_id	course_name	department	credits
101	SQL Basics	IT	3
104	Excel	Finance	2
105	Statistics	HR	3

8. SELECT ~~course~~ course_id,
course_name,
credits
FROM courses
~~WHERE credits~~
ORDER BY credits DESC
LIMIT 3;

course_id	course_name	credits
102	Python	4
103	Data Science	4
105	Statistics	3

9. SELECT ~~enrollment_id,~~
~~student_id,~~
~~course_id,~~
MAX(grade) AS max_grade,
MIN(grade) AS min_grade,
AVG(grade) AS avg_grade
FROM enrollments;
~~GROUP BY enrollment_id;~~

MAX_grade	Min_grad	avg_grade

10 SELECT course_id,
COUNT(enrollment_id) AS enrollment_count
FROM enrollments
GROUP BY course_id;

course_id	enrollment_count
101	1
102	1
103	1
104	1
105	1

11. SELECT department,
SUM(salary) AS total_salary,
SUM(bonus) AS total_bonus
FROM salaries
GROUP BY department; ✓✓

department	~~salary~~ total_salary	~~bonus~~ total_bonus
IT	122 000	11500
HR	109 000	7500
Finance	70 000	6000

12. SELECT department
AVG(salary) As avg_salary
FROM salaries
GROUP BY department
HAVING AVG(salary) > 55 000 ; ✓

department	avg_salary	
IT	61 000	✓
Finance	70 000	✓
~~HR~~	~~54500~~	✗ < 55000

13.
```
SELECT employee_id,
       name,
       salary,
       bonus,
       (salary + bonus) AS total_compensation
FROM salaries
WHERE ~~total_salaries~~ salary + bonus > 60 000;
```

emp_id	~~department~~ name	~~total_salary~~ salary	~~total~~ bonus	total_compensation
1	Tom	60000	5000	65 000
3	Spike	70 000	6000	76 000
4	Tyke	62 000	5500	67 500

Exercise 3: SQL CASE Statements

1. SELECT product_name,
price,
CASE
WHEN price < 100 THEN 'Budget'
WHEN price BETWEEN 100 AND 1000 THEN 'Mid-range'
WHEN price > 1000 THEN 'Expensive'
END AS price_category
FROM products;

product_name	price	price_category
Laptop	1200.00	Expensive
Phone	800.00	Mid-range
Keyboard	45.00	Budget
Monitor	300.00	Mid-range
Mouse	25.00	Budget

2. SELECT customer_name,
amount,
CASE
WHEN amount < 500 THEN 'Low Value'
WHEN amount BETWEEN 500 AND 999.99 THEN 'Medium Value'
ELSE 'High Value'
END AS order_value_category
FROM orders;

customer_name	amount	order_value_category
Alice	150.00	Low Value
Bob	560.00	Medium Value
Charlie	999.99	Medium Value
Diana	45.50	Low Value
Ethan	1200.00	High Value

3. SELECT emp_name,
department,
salary

CASE
WHEN department = 'IT' AND salary > 80 000
THEN 'Senior IT'
WHEN department = 'HR' AND salary > 55 000
THEN 'Experenced HR'
ELSE 'Staff'
END AS position_level
FROM employees;

4. SELECT student_name,
score,
CASE
WHEN score ≥ 90 THEN 'A'
WHEN score BETWEEN 80 AND 89 THEN 'B'
WHEN score BETWEEN 70 AND 79 THEN 'C'
WHEN score BETWEEN 60 AND 69 THEN 'D'
ELSE 'F'
END AS grade,
FROM students;

student_name	score	grade
Anna	92	A
Ben	76	C
Carg	59	F
David	83	B
Ella	68	D

5.
```
SELECT delivery_id,
       delivery_time_minutes
CASE
    WHEN delivery_time_minutes ≤ 30 THEN 'Fast'
    WHEN delivery_time_minutes BETWEEN 31 AND 60 THEN 'On
    ELSE 'Late'
END AS performance
FROM deliverres;
```

delivery_id	delivery_time_minutes	performance
1	45	On time
2	60	Late
3	30	Fast
4	65	LATE
5	100	LATE

6.
```
SELECT issue_type,
       priority
CASE
    WHEN priority = 3 THEN 'High'
    WHEN priority = 2 THEN 'Medium'
    WHEN priority = 1 THEN 'LOW'
END AS priority_label
FROM ~~issu~~ tickets;
```

7. SELECT student_id,
(days_present * 100 / total_days) AS attendance_percentage,
CASE
WHEN (days_present * 100 / total_days) ≥ 90
THEN 'Excellent'
WHEN (days_present * 100 / total_days) BETWEEN
75 AND 89 THEN 'Good'
ELSE 'Needs Improvement'
END AS attendance_status
FROM attendance;

8. SELECT product_id,
stock_qty,
CASE
WHEN stock_qty = 0 THEN 'Out of Stock'
WHEN stock_qty BETWEEN 1 AND 5 'Low Stock'
ELSE 'In Stock'
END AS stock_status
FROM products_inventory;

9.
```
SELECT subject,
       enrolled_students,
  CASE
    WHEN enrolled_students >= 25 THEN 'Large'
    WHEN enrolled_students BETWEEN 10 AND 24 THEN 'Medium'
    ELSE 'Small'
  END AS class_size_category
FROM classes;
```

subject	enrolled_students	class_size_category
MATH	30	Large
English	25	Large
Science	15	Medium
Art	5	Small
History	20	Medium

10.
```
SELECT payment_id,
       payment_method,
       amount,
  CASE
    WHEN payment_method = 'Cash' AND amount >= 200
    THEN 'Eligible for Discount'
    ELSE 'Not Eligible'
  END AS 'discount_eligibility'
FROM payments.
```

10. SELECT payment_id,
payment_method,
amount,
CASE
WHEN payment_method = 'Cash' AND amount ≥ 200
THEN 'Eligible for Discount'
ELSE 'Not Eligible'
END AS discount_eligibility
FROM payments;

payment_id	payment_method	amount	discount_eligibility
1	Card	50	Not eligible
2	cash	200	Eligible for discount
3	Card	150.00	Not eligible
4	Paypal	75.00	Not eligible
5	Cash	300.00	Eligible