SQL Basics Summary

SELECT Statement

```sql
SELECT * FROM customers;
SELECT first_name, country FROM customers;
```

INSERT Statement

```sql
INSERT INTO customers (id, first_name, country, score)
VALUES (6, 'Ali', 'Egypt', 400);
```

UPDATE Statement

```sql
UPDATE customers
SET score = 500
WHERE id = 6;
```

DELETE Statement

```sql
DELETE FROM customers
WHERE id = 6;
```

WHERE Clause

```sql
SELECT * FROM customers
WHERE country = 'USA';
```

Filtering Data

```sql
SELECT * FROM customers
WHERE score > 500 AND country = 'Germany';
```

Keyword / Operator	Definition	Example
WHERE	Filters rows based on a condition	WHERE country = 'USA'
=	Equal to	score = 500
!= or <>	Not equal to	country != 'USA'
>	Greater than	score > 500
<	Less than	score < 500
>=	Greater than or equal	score >= 500

<=	Less than or equal	score <= 500
AND	All conditions must be true	score > 500 AND country = 'Germany'
OR	At least one condition is true	country = 'USA' OR country = 'Germany'
IN	Matches values in a list	country IN ('USA', 'Germany')
BETWEEN	Value within a range (inclusive)	score BETWEEN 400 AND 800
LIKE	Pattern matching using %	first_name LIKE 'M%'

ORDER BY Clause

```
SELECT * FROM customers
ORDER BY score DESC;


SELECT * FROM customers
ORDER BY score ASC;
```

TOP Clause

```
SELECT TOP 3 * FROM customers;
```

HAVING – Filter After Aggregation

```
SELECT country, AVG(score) AS avg_score
FROM customers
GROUP BY country
HAVING AVG(score) > 430;
```

Aggregate Functions

- Aggregate functions perform calculations on a group of rows and return on single value
- There are usually used GROUP BY

Function	What it does	Example
COUNT()	Counts rows	COUNT(*)
SUM()	Adds values	SUM(sales)
AVG()	Calculates average	AVG(score)

MIN()	Finds smallest value	MIN(score)
MAX()	Finds largest value	MAX(score)

```sql
SELECT COUNT(*) AS total_customers
FROM customers;

SELECT SUM(sales) AS total_sales
FROM orders;

SELECT AVG(score) AS average_score
FROM customers;

SELECT
    MIN(score) AS lowest_score,
    MAX(score) AS highest_score
FROM customers;

SELECT country, AVG(score) AS avg_score
FROM customers
GROUP BY country;

SELECT country, AVG(score) AS avg_score
FROM customers
GROUP BY country
HAVING AVG(score) > 400;
```

INNER JOIN

```sql
SELECT c.first_name, o.sales
FROM customers c
INNER JOIN orders o
ON c.id = o.customer_id;
```

LEFT JOIN

```sql
SELECT c.first_name, o.sales
FROM customers c
LEFT JOIN orders o
ON c.id = o.customer_id;
```

UNION – Set Operator

```sql
SELECT country FROM customers
UNION
SELECT country FROM suppliers;
```

JOIN Type	What it returns	Example result
INNER JOIN	Only rows that match in both tables	Customers who have orders
LEFT JOIN	All rows from left table + matching rows from right	All customers, orders if they exist
RIGHT JOIN	All rows from right table + matching rows from left	All orders, customer info if exists
FULL JOIN	All rows from both tables	All customers and all orders
CROSS JOIN	Every row combined with every row	Cartesian product
SELF JOIN	A table joined with itself	Compare rows within same table

```sql
-- INNER
SELECT * FROM customers c INNER JOIN orders o ON c.id = o.customer_id;

-- LEFT
SELECT * FROM customers c LEFT JOIN orders o ON c.id = o.customer_id;

-- RIGHT
SELECT * FROM customers c RIGHT JOIN orders o ON c.id = o.customer_id;

-- FULL
SELECT * FROM customers c FULL JOIN orders o ON c.id = o.customer_id;

-- CROSS
SELECT * FROM customers CROSS JOIN orders;
```

Set Operator	What it does	Removes duplicates?	Condition
UNION	Combines results of two queries	✅ Yes	Same number & type of columns
UNION ALL	Combines results including duplicates	❌ No	Same number & type of columns
INTERSECT	Returns common rows from both queries	✅ Yes	Same structure
EXCEPT (MINUS in Oracle)	Returns rows from first query not in second	✅ Yes	Same structure

```sql
-- UNION
SELECT country FROM customers
UNION
SELECT country FROM suppliers;

-- UNION ALL
SELECT country FROM customers
UNION ALL
SELECT country FROM suppliers;

-- INTERSECT
SELECT country FROM customers
INTERSECT
SELECT country FROM suppliers;

-- EXCEPT
SELECT country FROM customers
EXCEPT
SELECT country FROM suppliers;
```

Constraints – Table Rules

Constraint	Purpose / Description	Rules / Notes	Example
PRIMARY KEY	Uniquely identifies each row in a table	Must be unique, cannot be NULL	id INT PRIMARY KEY
FOREIGN KEY	Creates a link between two tables (referential integrity)	Must match a primary key in another table	customer_id INT FOREIGN KEY REFERENCES customers(id)
NOT NULL	Column must have a value	Cannot be empty	first_name VARCHAR(50) NOT NULL
UNIQUE	Ensures all values in a column are different	No duplicates allowed	email VARCHAR(100) UNIQUE
DEFAULT	Provides a default value if none is given	Optional, only applies if value not provided	country VARCHAR(50) DEFAULT 'Egypt'
