

SQL 快速参考

SELECT、JOIN、子查询、索引与事务

SELECT

```
SELECT * FROM users;
SELECT name, email FROM users;
SELECT DISTINCT city FROM users;
SELECT name AS full_name FROM users;
```

WHERE

比较运算符

= <> (!=)	等于 / 不等于
< > <= >=	比较运算符
AND OR NOT	逻辑运算符
IS NULL / IS NOT NULL	Null 检查

模式匹配

```
SELECT * FROM users WHERE name LIKE 'A%';
-- % = any chars, _ = single char
SELECT * FROM users WHERE age IN (20, 25, 30);
SELECT * FROM users WHERE age BETWEEN 18 AND 30;
```

JOIN

JOIN 类型

INNER JOIN	两表中均匹配的行
LEFT JOIN	左表全部行 + 右表匹配行
RIGHT JOIN	右表全部行 + 左表匹配行
FULL OUTER JOIN	两表全部行
CROSS JOIN	两表的笛卡尔积

JOIN 语法

```
SELECT u.name, o.total
FROM users u
INNER JOIN orders o ON u.id = o.user_id;

SELECT u.name, o.total
FROM users u
LEFT JOIN orders o ON u.id = o.user_id;
```

INSERT / UPDATE / DELETE

插入

```
INSERT INTO users (name, email)
VALUES ('Alice', 'alice@example.com');

INSERT INTO users (name, email) VALUES
('Bob', 'bob@ex.com'),
('Carol', 'carol@ex.com');
```

更新

```
UPDATE users SET email = 'new@ex.com'
WHERE id = 1;
```

删除

```
DELETE FROM users WHERE id = 1;
DELETE FROM users; -- delete all rows
```

CREATE TABLE

语法

```
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  email TEXT UNIQUE,
  age INTEGER DEFAULT 0,
  score REAL
);
```

refmint.com

常用数据类型

INTEGER	整数
REAL	浮点数
TEXT	字符串 / 文本数据
BLOB	二进制数据
BOOLEAN	TRUE / FALSE (存储为 0/1)
DATE / DATETIME	日期与时间戳值

约束

PRIMARY KEY	唯一行标识符
NOT NULL	值必填
UNIQUE	不允许重复值
DEFAULT val	省略时的默认值
CHECK (expr)	自定义验证规则
FOREIGN KEY	引用另一张表

聚合函数

COUNT(*)	行数
COUNT(col)	列中的非 null 值数量
SUM(col)	数值列的总和
AVG(col)	数值列的平均值
MIN(col)	最小值
MAX(col)	最大值

示例

```
SELECT COUNT(*) AS total,
       AVG(age) AS avg_age,
       MAX(score) AS top_score
FROM users;
```

GROUP BY / HAVING

```
SELECT city, COUNT(*) AS num_users
FROM users
GROUP BY city;

SELECT city, AVG(age) AS avg_age
FROM users
GROUP BY city
HAVING AVG(age) > 25;
```

WHERE 在分组前过滤行; HAVING 在聚合后过滤分组

ORDER BY / LIMIT

```
SELECT * FROM users ORDER BY name ASC;
SELECT * FROM users ORDER BY age DESC;
SELECT * FROM users
ORDER BY city, name
LIMIT 10 OFFSET 20; -- skip 20, take 10
```

子查询

在 WHERE 子句中

```
SELECT name FROM users
WHERE id IN (
  SELECT user_id FROM orders
  WHERE total > 100
);
```

作为派生表

```
SELECT city, avg_age FROM (
  SELECT city, AVG(age) AS avg_age
  FROM users GROUP BY city
) WHERE avg_age > 30;
```

索引

```
CREATE INDEX idx_name ON users(name);
CREATE UNIQUE INDEX idx_email
  ON users(email);
DROP INDEX idx_name;
```

何时建索引

WHERE 条件列	加速过滤
JOIN ON 列	加速 JOIN 查找
ORDER BY 列	加速排序
高基数列	唯一值越多, 收益越大