



## 조건 검색 수행

### Database Fundamentals

발표자 이름

날짜

© 2019, Amazon Web Services, Inc. 또는 계열사. All rights reserved.

조건 검색 수행을 시작하겠습니다.

## 교육 내용

### 이 강의의 핵심

배울 내용은 다음과 같습니다.

- 하나 이상의 조건을 사용하여 검색
- 다양한 값 및 NULL 값 검색
- 문자열 패턴을 기반으로 데이터 검색

주요 용어:

- SELECT 스테이트먼트
- FROM 스테이트먼트
- WHERE 값
- 코멘트




이 모듈에서 학습할 내용은 다음과 같습니다.

- SELECT 스테이트먼트를 사용하여 데이터베이스에서 데이터 검색
- WHERE 절을 사용하여 테이블의 특정 행만 반환하도록 요청
- SELECT 스테이트먼트의 올바른 구문 파악
- 특정 열 또는 모든 열에서 데이터 선택

**SELECT** 스테이트먼트에는 사용할 수 있는 몇 가지 기본 절이 있으며, 필요한 유일한 절은 **FROM**입니다.

SELECT 스테이트먼트에는 선택할 수 있는 기본 절이 5개가 있으며, 필요한 유일한 절은 FROM입니다. SELECT 키워드는 결과에 어떤 열을 반환할지 결정하는 데 사용한다는 사실을 상기해 봅시다. 키워드 FROM은 쿼리가 수행되는 테이블을 지정하며, 검색을 수행하는 데 필요합니다.



단순 검색 조건

**검색 조건**은 행에 적용할 수 있는 논리적 테스트입니다. 이 구조는 2개의 값 표현식과 하나의 연산자를 사용하며, 두 값 간의 관계를 테스트합니다.

검색 조건은 각 행에 적용할 수 있는 논리적 테스트입니다. 2개의 값 표현식과 하나의 연산자의 형식을 취하며, 두 값 간의 관계를 테스트합니다.

## 검색 조건 예시

다음 **검색 조건**을 생각해 보십시오.

표현식  $\longrightarrow$  값\_A > 값\_B

↑  
연산자

## 단일 검색 조건


```
SELECT ordnum, sldate, qty, partnum, repid  
FROM sales  
WHERE repid = 'NO2'
```

테이블에서 선택한 값에 대해  
검색할 조건

열 이름

값

다음 예는 등호(=) 비교 연산자가 있는 WHERE 절을 사용하는 조건 검색을 보여줍니다. 검색 조건은 선택, 업데이트 또는 삭제할 행을 선택하는 기준을 지정합니다. 검색 조건은 절, 모든 DML 스테이트먼트의 표현식, 일부 DDL 스테이트먼트와 함께 사용되는 파라미터입니다.



연산자



## 연산자

**연산자**는 데이터 항목을 조작하여 결과를 반환

**연산자**는 다양한 정형 쿼리 언어(SQL) 연산에서 사용됩니다.

### **SELECT, INSERT, UPDATE 또는 DELETE**

**산술 연산자**는 숫자 피연산자에 대해 산술 연산을 수행합니다.

**비교 연산자**는 데이터베이스의 항목에 대해 비교 연산을 수행합니다. 주로 SQL에서 등식과 부등식을 테스트하는 데 사용됩니다.



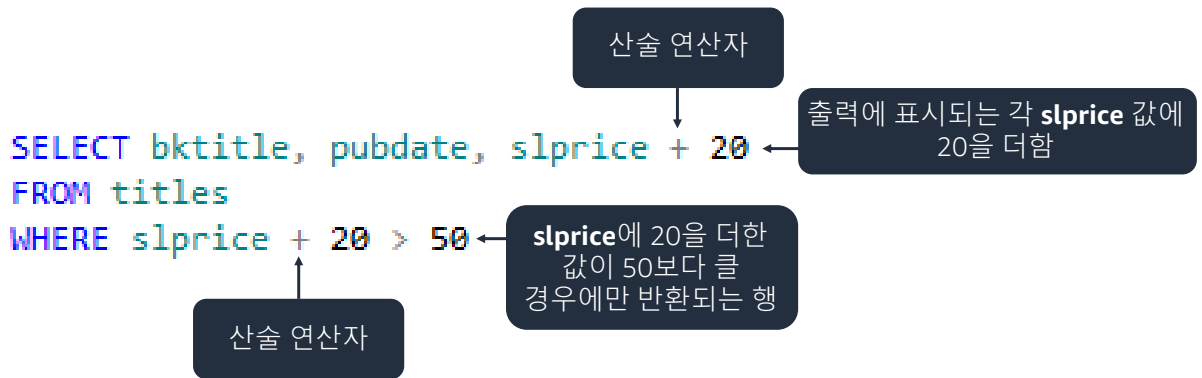
산술 연산자


## 산술 연산자

**a**에 5라는 값이 있고 **b**에도 5라는 값이 있다고 가정해 보겠습니다.

연산자	설명	예시
+(덧셈)	연산자의 한 측에 값을 더함	$a + b$ 는 10을 반환
-(뺄셈)	왼쪽 피연산자에서 오른쪽 피연산자를 뺌	$a - b$ 는 0을 반환
*(곱셈)	연산자의 한 측에 값을 곱함	$a * b$ 는 25를 반환
/(나눗셈)	왼쪽 피연산자를 오른쪽 피연산자로 나눔	$b / a$ 는 1을 반환
%(계수)	왼쪽 피연산자를 오른쪽 피연산자로 나누고 나머지를 반환	$b \% a$ 는 0을 반환

## 산술 연산자





비교 연산자

## 비교 연산자

**a**에 **5**라는 값이 있고 **b**에는 **10**이라는 값이 있다고 가정해 보겠습니다.

연산자	설명	예시
=	두 피연산자의 값이 동일한지 여부를 확인	(a = b)는 참이 아닙니다.
!=	두 피연산자의 값이 동일한지 여부를 확인	(a != b)는 참입니다.
<>	두 피연산자의 값이 동일한지 여부를 확인	(a <> b)는 참입니다.
>	왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 큰지 확인	(a > b)는 참이 아닙니다.
<	왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작은지 확인	(a < b)는 참입니다.

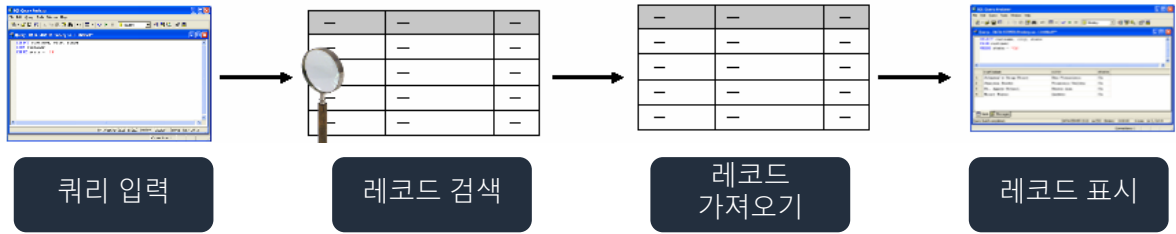
## WHERE 절: 예시 1

```
SELECT partnum, bkttitle, slprice  
FROM titles  
WHERE slprice > 100
```




다음 예는 > 연산자를 사용하는 WHERE 절을 보여 줍니다.

## 조건 검색 프로세스



다음 예는 조건 검색 흐름을 보여 줍니다.





논리 연산자

## 논리 연산자

논리 연산자	설명
ALL	모든 비교 집합이 TRUE인 경우 TRUE
AND	양 부울 표현식이 TRUE인 경우 TRUE
ANY	비교 집합 중 하나가 TRUE인 경우 TRUE
BETWEEN	피연산자가 범위 내에 있을 경우 TRUE
EXISTS	하위 쿼리에 행이 포함된 경우 TRUE
IN	피연산자가 표현식 목록 중 하나와 같은 경우 TRUE
LIKE	피연산자가 패턴과 일치할 경우 TRUE
OR	부울 표현식 중 하나가 TRUE일 경우 TRUE
SOME	비교 집합의 일부가 TRUE인 경우 TRUE
NOT	이 테이블에 나열된 이전 연산자에서 반환되는 논리적 값을 뒤집음

## WHERE 절: 예시 2

```
SELECT ordnum, sldate, qty, partnum, repid  
FROM sales  
WHERE repid = 'N02' AND qty >= 400
```



다음 예는 여러 검색 조건을 사용하는 WHERE 절을 보여 줍니다.

## 여러 조건 연산자

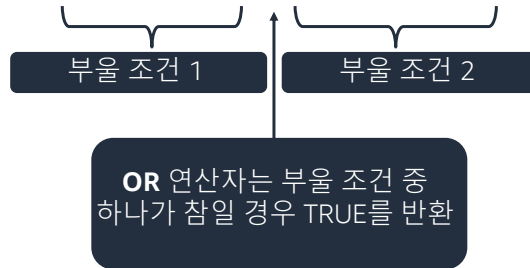
```
SELECT partnum, bktitle, slprice  
FROM titles  
WHERE slprice > 40 AND partnum > 1000
```



다음 조건 검색 예는 여러 조건 연산자를 사용합니다. 또한 논리 연산자인 AND를 사용합니다.

## OR 연산자

```
SELECT custnum, city, state, custname  
FROM customers  
WHERE state = 'MA' OR state = 'CA'
```



## AND 연산자

```
SELECT custnum, city, state  
FROM customers  
WHERE state = 'TX' AND city = 'Houston'
```



AND 연산자는 양 부울 조건이  
참일 경우 TRUE를 반환

## NOT 연산자

```
SELECT custnum, city, state, custname  
FROM customers  
WHERE NOT city = 'Ryebrook'
```

↑  
**NOT** 연산자는 조건의 결과를  
부정함

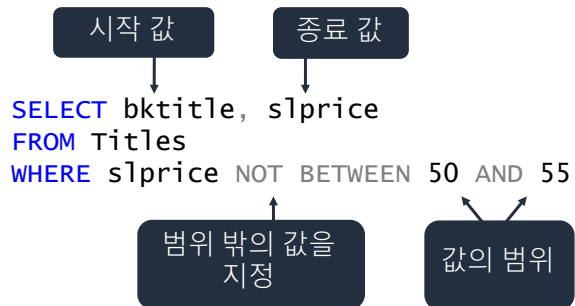
## AND 및 NOT 연산자의 조합

```
SELECT custnum, city, state, custname  
FROM customers  
WHERE state = 'NY' AND NOT city = 'Ryebrook'
```

AND와 함께 사용된 NOT



## BETWEEN...AND 연산자



두 비교 연산자를 사용하는 동등 **WHERE** 절

```
WHERE NOT (slprice >= 50 AND slprice <= 55)
```

## IN 연산자

```
SELECT custname  
FROM customers  
WHERE state IN ('CA' , 'NY' , 'MA')
```

열 이름

소괄호 내의 값 목록

IN 연산자는 값 목록과 일치하는 행을 반환

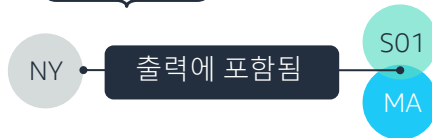
## 연산자 순위

연산자 수준	각 수준의 연산자 순위
1	+(긍정), -(부정), ~(비트 NOT)
2	*(곱셈), /(나눗셈), %(나머지)
3	+(덧셈), +(연결), -(뺄셈)
4	=, >, <, >=, <=, <>, !=, !=, !=(비교 연산자)
5	^(비트 간 배타적 OR), &(비트 AND),  (비트 OR)
6	NOT
7	AND
8	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
9	=(할당)

## AND 및 OR 조건에서 소괄호 사용

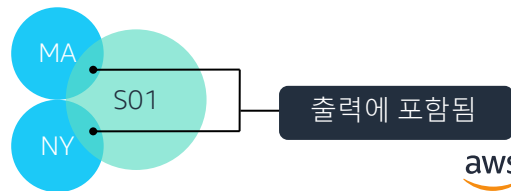
AND는 OR 전에  
수행됨

```
SELECT city, state, custname, repid  
FROM Customers  
WHERE state = 'NY' OR state = 'MA' AND repid = 'S01'
```



소괄호로  
재정의됨

```
SELECT city, state, custname, repid  
FROM Customers  
WHERE (state = 'NY' OR state = 'MA') AND repid = 'S01'
```



소괄호와 AND 절을 사용하여 **조건을 그룹화**합니다. 소괄호는 OR 비교를 분리하고 전체 블록을 비교하는 데 사용할 수도 있습니다.

## 활동: 연산자 사용



어떤 상황에서 쿼리에 연산자를 사용하겠습니까?  
정답을 다른 수강생과 함께 논의해 보십시오.

## 별칭

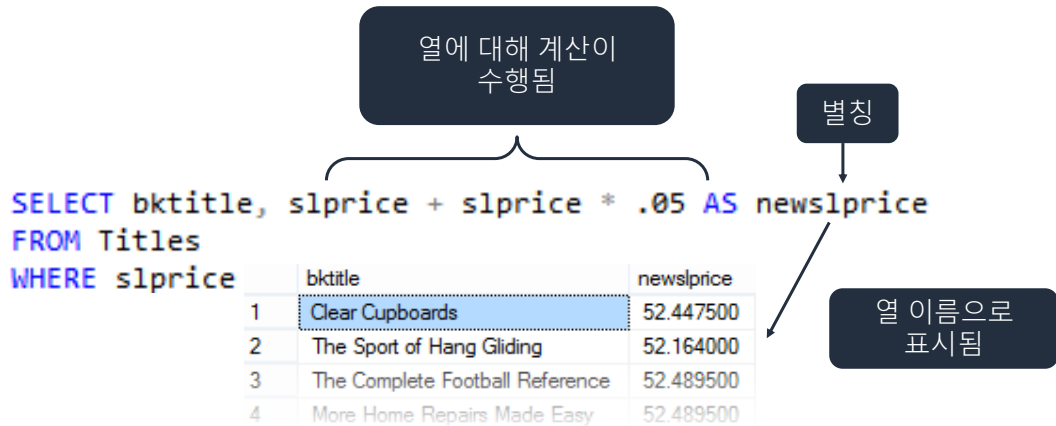
이전 예에서, 반환된 결과 집합은 데이터베이스의 열 이름 또는 필드 이름과 일치하는 열 이름을 사용했습니다. 이 섹션에서는 별칭을 사용하여 쿼리 결과를 반환합니다.

## 이름 지정 별칭 사용

```
SELECT col_name AS new_name
```

이름 지정 별칭은 원하는 열 머리글의 결과 집합을 제공하는 데 사용할 수 있습니다. 경우에 따라 별칭을 사용하면 SQL 스테이트먼트를 더 간단하게 쓰고 쉽게 읽을 수 있습니다. 열 이름 뒤에 AS 키워드를 사용하여 새 열 이름을 생성할 수 있습니다.

## 열 별칭 지정



이 예는 열 별칭 지정을 보여 줍니다.

열 별칭은 다음을 위해 사용됩니다.

- 열에 더 나은 이름 부여
- 자체 문서화
- SELECT 쿼리 중에 테이블 또는 열에 일시적으로 다른 이름 할당



## 별칭 사용 방법

AS를 사용하여 별칭 지정

```
SELECT bktitle, slprice + slprice * .05 AS newslprice
```

선택 사항인 AS 절이 생략됨

```
SELECT bktitle, slprice + slprice * .05 newslprice
```

여러 개의 단어 주위에 따옴표

```
SELECT bktitle, slprice + slprice * .05 AS 'New Sale Price'
```

별칭은 표현식과 같음

```
SELECT bktitle, newslprice = slprice + slprice * .05
```

별칭을 사용하여 SQL 스테이트먼트를 더 쉽게 읽고 쓸 수 있습니다.



NULL 값

## NULL 값

	bktitle	slprice	devcost	
1	Simple Auto Repairs	39.99	NULL	2개의 NULL 값이 같지 않음
2	Learning Japanese (Advanced)	30.00	NULL	
3	Conversational Japanese	35.00	NULL	
4	Learning Chinese (Advanced)	30.00	NULL	NULL과 같지 않음
5	Conversational Chinese	35.00	0.00	
6	Writing the Great American Novel	44.50	NULL	

값을 알 수 없음

단어 NULL이 표시됨

NULL 스테이트먼트는 누락된 값을 나타내는 데 사용된다는 사실을 상기해 봅시다. 테이블의 NULL 값은 필드에서 비어 있는 것으로 나타나는 값입니다.

## 비교에 NULL 값 사용

평가할 행

partnum	bktitle	devcost	slprice	pubdate
40891	Writing the Great American Novel	NULL	44.50	2017-05-25 00:00:00

쿼리

```
SELECT *
FROM Titles
WHERE slprice > 40 AND devcost > 15000
```

$44.50 > 40$        $AND$        $NULL > 15000$   
 TRUE       $AND$       NULL  
 FALSE

이 행은 결과에 포함되지 않음

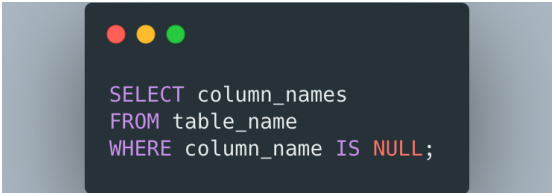
표현식 1	표현식 2	AND 연산의 결과	OR 연산의 결과
TRUE	NULL	FALSE	TRUE
NULL	NULL	FALSE	UNKNOWN
FALSE	NULL	FALSE	UNKNOWN

이 예는 비교에 NULL 값을 사용하는 방법을 보여 줍니다.

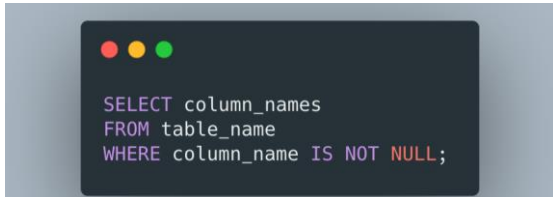
## NULL 값 테스트

NULL 값을 테스트할 때는 **IS NULL** 및 **IS NOT NULL**을 사용합니다.

- =, <, >와 같은 비교 연산자로 NULL 값을 테스트할 수 없습니다.
- NULL 값에 대해 테스트할 때는 IS NULL 및 IS NOT NULL 연산자를 대신 사용합니다.



```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```



```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

=, <, >와 같은 비교 연산자로 NULL 값을 테스트할 수 없습니다. NULL 값에 대해 테스트할 때는 IS NULL 및 IS NOT NULL을 사용합니다.

## IS NULL 절

쿼리

```
SELECT bktitle, slprice, devcost  
FROM titles  
WHERE slprice BETWEEN 35 AND 45 AND devcost IS NULL
```

조건 1  
이 범위의 slprice

두 조건이 모두  
참이어야 함

조건 2  
devcost에 NULL이  
포함됨

출력

	bktitle	slprice	devcost
1	Simple Auto Repairs	39.99	NULL
2	Conversational Japanese	35.00	NULL
3	Writing the Great American Novel	44.50	NULL

NULL 값을  
포함하는 행

## IS NOT NULL 절

쿼리

```
SELECT bktitle, slprice, devcost
FROM titles
WHERE slprice BETWEEN 35 AND 45 AND devcost IS NOT NULL
```

이 범위의 slprice

두 조건이 모두  
참이어야 함

devcost는 NULL이 아님

출력

	bktitle	slprice	devcost
1	The Complete Auto Repair Guide	50.99	16022.49
2	North American History	50.00	16349.94

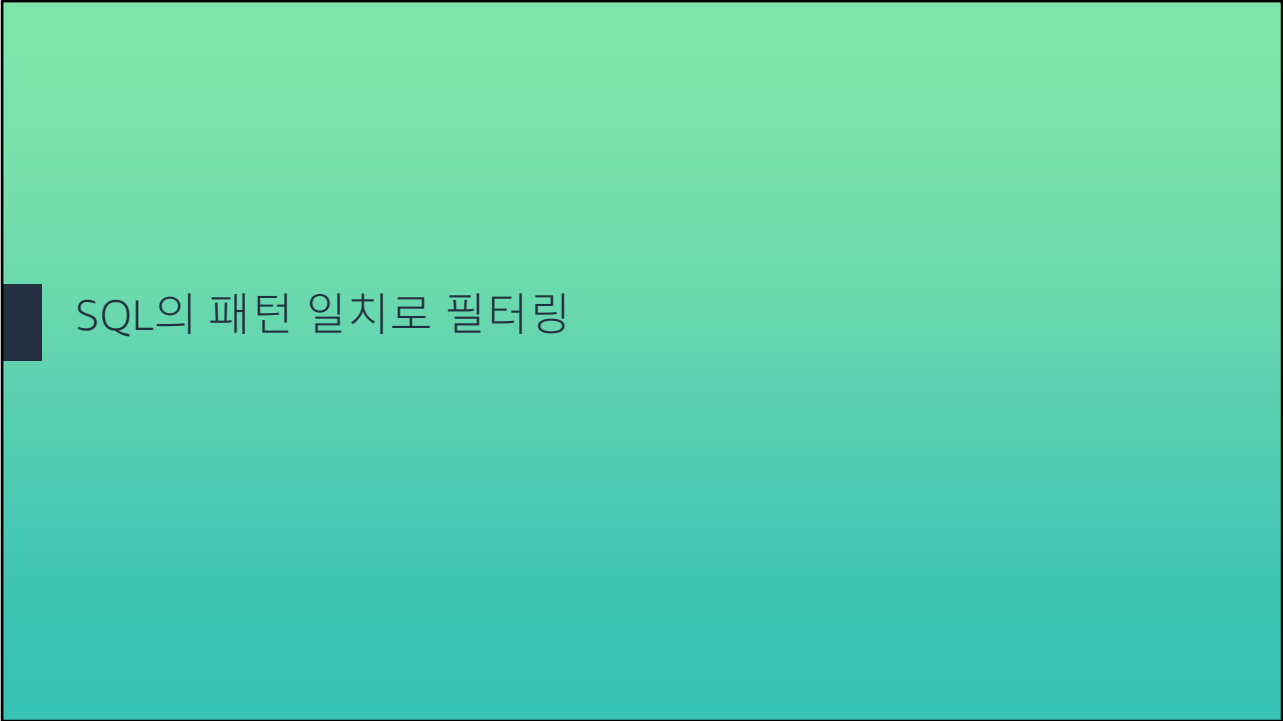
NULL 값을 포함하지  
않는 행

## 논의: NULL 값



1. NULL 값의 목적은 무엇입니까?
2. 테이블에서 NULL 값을 검색하는 것이 중요한 이유는 무엇입니까?
3. 테이블에서 NULL을 포함하지 않는 결과를 어떻게 검색하겠습니까?





SQL의 패턴 일치로 필터링

## 패턴 일치

아래에 나타난 **titles** 테이블을 고려할 때 다음 쿼리의 결과는 무엇입니까?

```
SELECT bktitle, slprice, partnum  
FROM titles  
WHERE bktitle LIKE '%art%'
```

```
SELECT bktitle, slprice, partnum  
FROM titles  
WHERE lowerof(bktitle) LIKE '%art%'
```

패턴

패턴

	bktitle	slprice	partnum
1	Starting a Small Garden	35.00	40321
2	Starting a Greenhouse	30.50	40322
3	The Art of Water Painting	34.50	40551
4	The Art of Oil Painting	34.50	40552
5	The Art of Pen and Ink Drawing	34.50	40553

출력

## 패턴 일치(계속)

→ `SELECT upperof(bktitle), slprice, partnum`  
`FROM titles`  
`WHERE bktitle LIKE '%art%'`

패턴

	bktitle	slprice	partnum
1	STARTING A SMALL GARDEN	35.00	40321
2	STARTING A GREENHOUSE	30.50	40322
3	THE ART OF WATER PAINTING	34.50	40551
4	THE ART OF OIL PAINTING	34.50	40552
5	THE ART OF PEN AND INK DRAWING	34.50	40553

출력

## upperof, lowerof

### SELECT에서 upperof와 lowerof의 목적

- 모두 대문자 또는 소문자로 결과 출력

### WHERE에서 upperof와 lowerof의 목적

- bktitle 열에서 art의 결과를 소문자로 출력

```
SELECT bktitle, slprice, partnum
FROM titles
WHERE lowerof(bktitle) LIKE
'%art%'
```

## 와일드카드

### 와일드카드의 목적

- %는 필드에서 하나 이상의 문자를 대체하는 것을 허용
- \_는 구문 표현식에서 단일 문자를 대체하는 것을 허용

- **WHERE** 절과 **LIKE** 연산자와 함께 사용됨
- 조합으로 사용할 수 있음

```
SELECT bktitle, slprice, partnum  
FROM titles  
WHERE bktitle LIKE 'c%'
```

열 이름

LIKE 연산자

와일드카드가 뒤에 따라오는 c 문자

이 표현식은 c로 시작하는 모든 책 제목 값을 찾는 것을 의미합니다.

## 여러 와일드카드

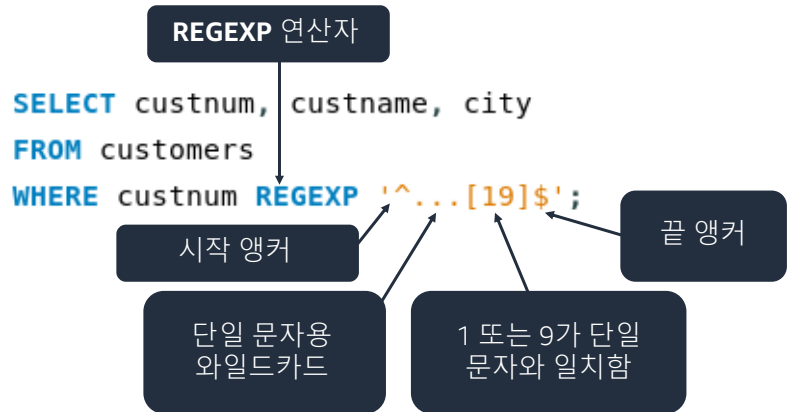
```
SELECT address, custname, state, city  
FROM Customers  
WHERE custname LIKE '_o%'
```

단일 문자용  
와일드카드

무제한 수의 문자용  
와일드카드

## 정규 표현식

- 지정된 텍스트에서 정보 필터링
- 2가지 유형:
  - 기본 정규 표현식(BRE)
  - 확장된 정규 표현식(ERE)



## 학습 내용 확인 질문



쿼리에서 조건의 목적은 무엇입니까?



SQL에서 별칭을 사용해야 하는 이유는 무엇입니까?

정답:

1. 쿼리에서 조건의 목적은 검색을 필터링할 수 있도록 하는 것입니다.
2. 별칭은 열 이름을 더 잘 이해할 수 있도록 만들기 위해 종종 사용됩니다.



# 핵심 사항



© 2020, Amazon Web Services, Inc. 또는 계열사. All rights reserved.

49

- **검색 조건**은 행에 적용할 수 있는 논리적 테스트입니다.
  - 구조는 2개의 값 표현식과 하나의 연산자를 사용합니다.
  - 두 값 간에 관계를 테스트합니다.
- 이름 지정 **별칭**은 원하는 열 머리글의 결과 집합을 제공합니다.
  - 경우에 따라 별칭을 사용하면 SQL 스테이트먼트를 더 간단하게 쓰고 쉽게 읽을 수 있습니다.

aws re/start

이 강의에서 다룬 핵심 사항은 다음과 같습니다.

- 검색 조건은 행에 적용할 수 있는 논리적 테스트입니다.
  - 구조는 2개의 값 표현식과 하나의 연산자를 사용합니다.
  - 두 값 간에 관계를 테스트합니다.
- 이름 지정 별칭은 원하는 열 머리글의 결과 집합을 제공합니다.
  - 경우에 따라 별칭을 사용하면 SQL 스테이트먼트를 더 간단하게 쓰고 쉽게 읽을 수 있습니다.