# [ 1 ]

**Query:** What is the total surface area of the countries in the Caribbean region?

# Schema:

# city(ID, Name, CountryCode, District, Population); # sqlite\_sequence(name, seq); # country(Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, Code2);

# countrylanguage(CountryCode, Language, IsOfficial, Percentage);

1. **Zero-shot prompt**

Schema:

# city(ID, Name, CountryCode, District, Population);

# sqlite\_sequence(name, seq);

# country(Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, Code2);

# countrylanguage(CountryCode, Language, IsOfficial, Percentage);

Query: 카리브해 지역의 국가들의 총 면적은 얼마인가?

SQL:

SELECT SUM(SurfaceArea)

FROM country

WHERE Region = 'Caribbean';

1. **3-shot prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# city(ID, Name, CountryCode, District, Population);

# sqlite\_sequence(name, seq);

# country(Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, Code2);

# countrylanguage(CountryCode, Language, IsOfficial, Percentage);

Query: 카리브해 지역의 국가들의 총 면적은 얼마인가?

SQL:

SELECT SUM(SurfaceArea)

FROM country

WHERE Region = 'Caribbean';

1. **Zero-shot Chain-of-Thought Prompt**

Schema:

# city(ID, Name, CountryCode, District, Population);

# sqlite\_sequence(name, seq);

# country(Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, Code2);

# countrylanguage(CountryCode, Language, IsOfficial, Percentage);

Query: 카리브해 지역의 국가들의 총 면적은 얼마인가?

단계별로:

1. 국가 정보가 포함된 테이블을 식별합니다: `country`.

2. `country` 테이블에서 관련된 열을 찾습니다: `Region` 및 `SurfaceArea`.

3. 'Caribbean' 지역으로 국가를 필터링합니다.

4. 필터링된 국가들의 `SurfaceArea`를 합산합니다.

SQL:

SELECT SUM(SurfaceArea)

FROM country

WHERE Region = 'Caribbean';

1. **3-shot Chain-of-Thought Prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# city(ID, Name, CountryCode, District, Population);

# sqlite\_sequence(name, seq);

# country(Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, Code2);

# countrylanguage(CountryCode, Language, IsOfficial, Percentage);

Query: 카리브해 지역의 국가들의 총 면적은 얼마인가?

단계별로:

1. 국가 정보가 포함된 테이블을 식별합니다: `country`.

2. `country` 테이블에서 관련된 열을 찾습니다: `Region` 및 `SurfaceArea`.

3. 'Caribbean' 지역으로 국가를 필터링합니다.

4. 필터링된 국가들의 `SurfaceArea`를 합산합니다.

SQL:

SELECT SUM(SurfaceArea)

FROM country

WHERE Region = 'Caribbean';

# [ 2 ]

**Query:** What are the record companies of orchestras in descending order of years in which they were founded?

# Schema:

# conductor(Conductor\_ID, Name, Age, Nationality, Year\_of\_Work);

# orchestra(Orchestra\_ID, Orchestra, Conductor\_ID, Record\_Company, Year\_of\_Founded, Major\_Record\_Format);

# performance(Performance\_ID, Orchestra\_ID, Type, Date, Official\_ratings\_(millions), Weekly\_rank, Share);

# show(Show\_ID, Performance\_ID, If\_first\_show, Result, Attendance);

1. **Zero-shot prompt**

Schema:

# conductor(Conductor\_ID, Name, Age, Nationality, Year\_of\_Work);

# orchestra(Orchestra\_ID, Orchestra, Conductor\_ID, Record\_Company, Year\_of\_Founded, Major\_Record\_Format);

# performance(Performance\_ID, Orchestra\_ID, Type, Date, Official\_ratings\_(millions), Weekly\_rank, Share);

# show(Show\_ID, Performance\_ID, If\_first\_show, Result, Attendance);

Query: 오케스트라의 설립 연도에 따른 음반사의 내림차순 정렬은 무엇인가?

SQL:

SELECT Record\_Company

FROM orchestra

ORDER BY Year\_of\_Founded DESC;

1. **3-shot prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# conductor(Conductor\_ID, Name, Age, Nationality, Year\_of\_Work);

# orchestra(Orchestra\_ID, Orchestra, Conductor\_ID, Record\_Company, Year\_of\_Founded, Major\_Record\_Format);

# performance(Performance\_ID, Orchestra\_ID, Type, Date, Official\_ratings\_(millions), Weekly\_rank, Share);

# show(Show\_ID, Performance\_ID, If\_first\_show, Result, Attendance);

Query: 오케스트라의 설립 연도에 따른 음반사의 내림차순 정렬은 무엇인가?

SQL:

SELECT Record\_Company

FROM orchestra

ORDER BY Year\_of\_Founded DESC;

1. **Zero-shot Chain-of-Thought Prompt**

Schema:

# conductor(Conductor\_ID, Name, Age, Nationality, Year\_of\_Work);

# orchestra(Orchestra\_ID, Orchestra, Conductor\_ID, Record\_Company, Year\_of\_Founded, Major\_Record\_Format);

# performance(Performance\_ID, Orchestra\_ID, Type, Date, Official\_ratings\_(millions), Weekly\_rank, Share);

# show(Show\_ID, Performance\_ID, If\_first\_show, Result, Attendance);

Query: 오케스트라의 설립 연도에 따른 음반사의 내림차순 정렬은 무엇인가?

단계별로:

1. 오케스트라 정보가 포함된 테이블을 식별합니다: `orchestra`.

2. `orchestra` 테이블에서 관련된 열을 찾습니다: `Record\_Company` 및 `Year\_of\_Founded`.

3. `Year\_of\_Founded`에 따라 내림차순으로 정렬합니다.

SQL:

SELECT Record\_Company

FROM orchestra

ORDER BY Year\_of\_Founded DESC;

1. **3-shot Chain-of-Thought Prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# conductor(Conductor\_ID, Name, Age, Nationality, Year\_of\_Work);

# orchestra(Orchestra\_ID, Orchestra, Conductor\_ID, Record\_Company, Year\_of\_Founded, Major\_Record\_Format);

# performance(Performance\_ID, Orchestra\_ID, Type, Date, Official\_ratings\_(millions), Weekly\_rank, Share);

# show(Show\_ID, Performance\_ID, If\_first\_show, Result, Attendance);

Query: 오케스트라의 설립 연도에 따른 음반사의 내림차순 정렬은 무엇인가?

단계별로:

1. 오케스트라 정보가 포함된 테이블을 식별합니다: `orchestra`.

2. `orchestra` 테이블에서 관련된 열을 찾습니다: `Record\_Company` 및 `Year\_of\_Founded`.

3. `Year\_of\_Founded`에 따라 내림차순으로 정렬합니다.

SQL:

SELECT Record\_Company

FROM orchestra

ORDER BY Year\_of\_Founded DESC;

# [ 3 ]

**Query:** Which region is the city Kabul located in?

# Schema:

# Breeds(breed\_code, breed\_name);

# Charges(charge\_id, charge\_type, charge\_amount); # Sizes(size\_code, size\_description);

# Treatment\_Types(treatment\_type\_code, treatment\_type\_description);

# Owners(owner\_id, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Dogs(dog\_id, owner\_id, abandoned\_yn, breed\_code, size\_code, name, age, date\_of\_birth, gender, weight, date\_arrived, date\_adopted, date\_departed);

# Professionals(professional\_id, role\_code, first\_name, street, city, state, zip\_code, last\_name, email\_address, home\_phone, cell\_number);

# Treatments(treatment\_id, dog\_id, professional\_id, treatment\_type\_code, date\_of\_treatment, cost\_of\_treatment);

1. **Zero-shot prompt**

Schema:

# Breeds(breed\_code, breed\_name);

# Charges(charge\_id, charge\_type, charge\_amount);

# Sizes(size\_code, size\_description);

# Treatment\_Types(treatment\_type\_code, treatment\_type\_description);

# Owners(owner\_id, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Dogs(dog\_id, owner\_id, abandoned\_yn, breed\_code, size\_code, name, age, date\_of\_birth, gender, weight, date\_arrived, date\_adopted, date\_departed);

# Professionals(professional\_id, role\_code, first\_name, street, city, state, zip\_code, last\_name, email\_address, home\_phone, cell\_number);

# Treatments(treatment\_id, dog\_id, professional\_id, treatment\_type\_code, date\_of\_treatment, cost\_of\_treatment);

Query: 카불 도시가 속한 지역은 어디인가?

SQL:

SELECT state

FROM Owners

WHERE city = 'Kabul';

1. **3-shot prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# Breeds(breed\_code, breed\_name);

# Charges(charge\_id, charge\_type, charge\_amount);

# Sizes(size\_code, size\_description);

# Treatment\_Types(treatment\_type\_code, treatment\_type\_description);

# Owners(owner\_id, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Dogs(dog\_id, owner\_id, abandoned\_yn, breed\_code, size\_code, name, age, date\_of\_birth, gender, weight, date\_arrived, date\_adopted, date\_departed);

# Professionals(professional\_id, role\_code, first\_name, street, city, state, zip\_code, last\_name, email\_address, home\_phone, cell\_number);

# Treatments(treatment\_id, dog\_id, professional\_id, treatment\_type\_code, date\_of\_treatment, cost\_of\_treatment);

Query: 카불 도시가 속한 지역은 어디인가?

SQL:

SELECT state

FROM Owners

WHERE city = 'Kabul';

1. **Zero-shot Chain-of-Thought Prompt**

Schema:

# Breeds(breed\_code, breed\_name);

# Charges(charge\_id, charge\_type, charge\_amount);

# Sizes(size\_code, size\_description);

# Treatment\_Types(treatment\_type\_code, treatment\_type\_description);

# Owners(owner\_id, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Dogs(dog\_id, owner\_id, abandoned\_yn, breed\_code, size\_code, name, age, date\_of\_birth, gender, weight, date\_arrived, date\_adopted, date\_departed);

# Professionals(professional\_id, role\_code, first\_name, street, city, state, zip\_code, last\_name, email\_address, home\_phone, cell\_number);

# Treatments(treatment\_id, dog\_id, professional\_id, treatment\_type\_code, date\_of\_treatment, cost\_of\_treatment);

Query: 카불 도시가 속한 지역은 어디인가?

단계별로:

1. 도시 정보가 포함된 테이블을 식별합니다: `Owners`.

2. `Owners` 테이블에서 관련된 열을 찾습니다: `city` 및 `state`.

3. `city`가 'Kabul'인 행을 필터링합니다.

4. 해당 행의 `state` 값을 반환합니다.

SQL:

SELECT state

FROM Owners

WHERE city = 'Kabul';

1. **3-shot Chain-of-Thought Prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# Breeds(breed\_code, breed\_name);

# Charges(charge\_id, charge\_type, charge\_amount);

# Sizes(size\_code, size\_description);

# Treatment\_Types(treatment\_type\_code, treatment\_type\_description);

# Owners(owner\_id, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Dogs(dog\_id, owner\_id, abandoned\_yn, breed\_code, size\_code, name, age, date\_of\_birth, gender, weight, date\_arrived, date\_adopted, date\_departed);

# Professionals(professional\_id, role\_code, first\_name, street, city, state, zip\_code, last\_name, email\_address, home\_phone, cell\_number);

# Treatments(treatment\_id, dog\_id, professional\_id, treatment\_type\_code, date\_of\_treatment, cost\_of\_treatment);

Query: 카불 도시가 속한 지역은 어디인가?

단계별로:

1. 도시 정보가 포함된 테이블을 식별합니다: `Owners`.

2. `Owners` 테이블에서 관련된 열을 찾습니다: `city` 및 `state`.

3. `city`가 'Kabul'인 행을 필터링합니다.

4. 해당 행의 `state` 값을 반환합니다.

SQL:

SELECT state

FROM Owners

WHERE city

**[ 4 ]**

**Query:** How many friends does each student have?

# Schema:

# Highschooler(ID, name, grade); # Friend(student\_id, friend\_id);

# Likes(student\_id, liked\_id);

1. **Zero-shot prompt**

Schema:

# Highschooler(ID, name, grade);

# Friend(student\_id, friend\_id);

# Likes(student\_id, liked\_id);

Query: 각 학생이 몇 명의 친구를 가지고 있는가?

SQL:

SELECT student\_id, COUNT(friend\_id) AS num\_friends

FROM Friend

GROUP BY student\_id;

1. **3-shot prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# Highschooler(ID, name, grade);

# Friend(student\_id, friend\_id);

# Likes(student\_id, liked\_id);

Query: 각 학생이 몇 명의 친구를 가지고 있는가?

SQL:

SELECT student\_id, COUNT(friend\_id) AS num\_friends

FROM Friend

GROUP BY student\_id;

1. **Zero-shot Chain-of-Thought Prompt**

Schema:

# Highschooler(ID, name, grade);

# Friend(student\_id, friend\_id);

# Likes(student\_id, liked\_id);

Query: 각 학생이 몇 명의 친구를 가지고 있는가?

단계별로:

1. 친구 관계를 나타내는 테이블을 식별합니다: `Friend`.

2. `Friend` 테이블에서 관련된 열을 찾습니다: `student\_id` 및 `friend\_id`.

3. 각 `student\_id`별로 `friend\_id`의 수를 셉니다.

SQL:

SELECT student\_id, COUNT(friend\_id) AS num\_friends

FROM Friend

GROUP BY student\_id;

1. **3-shot Chain-of-Thought Prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

**[ 5 ]**

**Query:** What is the last name of the student who has a cat that is 3 years old?

# Schema:

# Student(StuID, LName, Fname, Age, Sex, Major, Advisor, city\_code); # Has\_Pet(StuID, PetID);

# Pets(PetID, PetType, pet\_age, weight);

1. **Zero-shot prompt**

Schema:

# Student(StuID, LName, Fname, Age, Sex, Major, Advisor, city\_code);

# Has\_Pet(StuID, PetID);

# Pets(PetID, PetType, pet\_age, weight);

Query: 3살짜리 고양이를 가진 학생의 성은 무엇인가?

SQL:

SELECT S.LName

FROM Student S

JOIN Has\_Pet HP ON S.StuID = HP.StuID

JOIN Pets P ON HP.PetID = P.PetID

WHERE P.PetType = 'Cat' AND P.pet\_age = 3;

1. **3-shot prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# Student(StuID, LName, Fname, Age, Sex, Major, Advisor, city\_code);

# Has\_Pet(StuID, PetID);

# Pets(PetID, PetType, pet\_age, weight);

Query: 3살짜리 고양이를 가진 학생의 성은 무엇인가?

SQL:

SELECT S.LName

FROM Student S

JOIN Has\_Pet HP ON S.StuID = HP.StuID

JOIN Pets P ON HP.PetID = P.PetID

WHERE P.PetType = 'Cat' AND P.pet\_age = 3;

1. **Zero-shot Chain-of-Thought Prompt**

Schema:

# Student(StuID, LName, Fname, Age, Sex, Major, Advisor, city\_code);

# Has\_Pet(StuID, PetID);

# Pets(PetID, PetType, pet\_age, weight);

Query: 3살짜리 고양이를 가진 학생의 성은 무엇인가?

단계별로:

1. `Pets` 테이블에서 3살짜리 고양이를 찾습니다.

2. 해당 고양이의 `PetID`를 `Has\_Pet` 테이블과 조인합니다.

3. `Has\_Pet` 테이블을 통해 학생의 `StuID`를 얻습니다.

4. `Student` 테이블을 `StuID`로 조인하여 해당 학생의 성(last name)을 찾습니다.

SQL:

SELECT S.LName

FROM Student S

JOIN Has\_Pet HP ON S.StuID = HP.StuID

JOIN Pets P ON HP.PetID = P.PetID

WHERE P.PetType = 'Cat' AND P.pet\_age = 3;

1. **3-shot Chain-of-Thought Prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했는가?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있는가?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# Highschooler(ID, name, grade);

# Friend(student\_id, friend\_id);

# Likes(student\_id, liked\_id);

Query: 각 학생이 몇 명의 친구를 가지고 있는가?

단계별로:

1. 친구 관계를 나타내는 테이블을 식별합니다: `Friend`.

2. `Friend` 테이블에서 관련된 열을 찾습니다: `student\_id` 및 `friend\_id`.

3. 각 `student\_id`별로 `friend\_id`의 수를 셉니다.

SQL:

SELECT student\_id, COUNT(friend\_id) AS num\_friends

FROM Friend

GROUP BY student\_id;

**[ 6 ]**

**Query:** Find the number of visitors who did not visit any museum opened after 2010.

# Schema:

# museum(Museum\_ID, Name, Num\_of\_Staff, Open\_Year); # visitor(ID, Name, Level\_of\_membership, Age);

# visit(Museum\_ID, visitor\_ID, Num\_of\_Ticket, Total\_spent);

1. **Zero-shot prompt**

Schema:

# museum(Museum\_ID, Name, Num\_of\_Staff, Open\_Year);

# visitor(ID, Name, Level\_of\_membership, Age);

# visit(Museum\_ID, visitor\_ID, Num\_of\_Ticket, Total\_spent);

Query: 2010년 이후에 개장한 박물관을 방문하지 않은 방문자의 수는 몇 명인가요?

SQL:

SELECT COUNT(DISTINCT V.visitor\_ID) AS num\_visitors\_not\_visited

FROM visitor V

WHERE V.ID NOT IN (

SELECT DISTINCT visit.visitor\_ID

FROM visit

JOIN museum ON visit.Museum\_ID = museum.Museum\_ID

WHERE museum.Open\_Year > 2010

);

1. **3-shot prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가요?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했나요?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있나요?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# museum(Museum\_ID, Name, Num\_of\_Staff, Open\_Year);

# visitor(ID, Name, Level\_of\_membership, Age);

# visit(Museum\_ID, visitor\_ID, Num\_of\_Ticket, Total\_spent);

Query: 2010년 이후에 개장한 박물관을 방문하지 않은 방문자의 수는 몇 명인가요?

단계별로:

1. 2010년 이후에 개장한 박물관의 목록을 찾습니다.

2. 해당 박물관을 방문한 방문자들의 목록을 찾습니다.

3. 방문자 정보가 있는 `visitor` 테이블과 방문 정보가 있는 `visit` 테이블을 조인합니다.

4. 2010년 이후에 개장한 박물관을 방문하지 않은 방문자들의 수를 계산합니다.

SQL:

SELECT COUNT(DISTINCT V.visitor\_ID) AS num\_visitors\_not\_visited

FROM visitor V

WHERE V.ID NOT IN (

SELECT DISTINCT visit.visitor\_ID

FROM visit

JOIN museum ON visit.Museum\_ID = museum.Museum\_ID

WHERE museum.Open\_Year > 2010

);

1. **Zero-shot Chain-of-Thought Prompt**

Schema:

# Breeds(breed\_code, breed\_name);

# Charges(charge\_id, charge\_type, charge\_amount);

# Sizes(size\_code, size\_description);

# Treatment\_Types(treatment\_type\_code, treatment\_type\_description);

# Owners(owner\_id, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Dogs(dog\_id, owner\_id, abandoned\_yn, breed\_code, size\_code, name, age, date\_of\_birth, gender, weight, date\_arrived, date\_adopted, date\_departed);

# Professionals(professional\_id, role\_code, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Treatments(treatment\_id, dog\_id, professional\_id, treatment\_type\_code, date\_of\_treatment, cost\_of\_treatment);

Query: Kabul 시에 위치한 도시는 어느 지역에 속해 있나요?

단계별로:

1. Kabul의 위치 정보가 있는 테이블을 식별합니다.

2. Kabul의 위치 정보를 얻기 위해 관련된 열을 찾습니다.

3. Kabul의 위치 정보를 제공하는 열에서 지역 정보를 찾습니다.

SQL: SELECT Region

FROM city

JOIN country ON city.CountryCode = country.Code

WHERE city.Name = 'Kabul';

Zero-shot Chain-of-Thought 프롬프트 예시입니다.

1. **3-shot Chain-of-Thought Prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가요?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했나요?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있나요?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# Breeds(breed\_code, breed\_name);

# Charges(charge\_id, charge\_type, charge\_amount);

# Sizes(size\_code, size\_description);

# Treatment\_Types(treatment\_type\_code, treatment\_type\_description);

# Owners(owner\_id, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Dogs(dog\_id, owner\_id, abandoned\_yn, breed\_code, size\_code, name, age, date\_of\_birth, gender, weight, date\_arrived, date\_adopted, date\_departed);

# Professionals(professional\_id, role\_code, first\_name, last\_name, street, city, state, zip\_code, email\_address, home\_phone, cell\_number);

# Treatments(treatment\_id, dog\_id, professional\_id, treatment\_type\_code, date\_of\_treatment, cost\_of\_treatment);

Query: Kabul 시에 위치한 도시는 어느 지역에 속해 있나요?

단계별로:

1. Kabul의 위치 정보가 있는 테이블을 식별합니다.

2. Kabul의 위치 정보를 얻기 위해 관련된 열을 찾습니다.

3. Kabul의 위치 정보를 제공하는 열에서 지역 정보를 찾습니다.

SQL: SELECT Region

FROM city

JOIN country ON city.CountryCode = country.Code

WHERE city.Name = 'Kabul';

**[ 7 ]**

**Query:** How many United Airlines flights go to City 'Aberdeen'?

# Schema:

# airlines(uid, Airline, Abbreviation, Country);

# airports(City, AirportCode, AirportName, Country, CountryAbbrev); # flights(Airline, FlightNo, SourceAirport, DestAirport);

1. **Zero-shot prompt**

Schema:

# airlines(uid, Airline, Abbreviation, Country);

# airports(City, AirportCode, AirportName, Country, CountryAbbrev);

# flights(Airline, FlightNo, SourceAirport, DestAirport);

Query: How many United Airlines flights go to City 'Aberdeen'?

SQL:

SELECT COUNT(\*)

FROM flights F

JOIN airlines A ON F.Airline = A.Airline

JOIN airports ADest ON F.DestAirport = ADest.AirportCode

WHERE A.Airline = 'United Airlines' AND ADest.City = 'Aberdeen';

1. **3-shot prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가요?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했나요?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있나요?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# airlines(uid, Airline, Abbreviation, Country);

# airports(City, AirportCode, AirportName, Country, CountryAbbrev);

# flights(Airline, FlightNo, SourceAirport, DestAirport);

Query: How many United Airlines flights go to City 'Aberdeen'?

단계별로:

1. United Airlines의 항공편 정보를 찾기 위해 관련 테이블을 식별합니다: `flights`와 `airlines`.

2. United Airlines의 항공편 목록을 찾습니다.

3. 목적지가 Aberdeen인 항공편의 수를 세어줍니다.

SQL:

SELECT COUNT(\*)

FROM flights F

JOIN airlines A ON F.Airline = A.Airline

JOIN airports ADest ON F.DestAirport = ADest.AirportCode

WHERE A.Airline = 'United Airlines' AND ADest.City = 'Aberdeen';

1. **Zero-shot Chain-of-Thought Prompt**

Schema:

# airlines(uid, Airline, Abbreviation, Country);

# airports(City, AirportCode, AirportName, Country, CountryAbbrev);

# flights(Airline, FlightNo, SourceAirport, DestAirport);

Query: How many United Airlines flights go to City 'Aberdeen'?

단계별로:

1. United Airlines의 항공편 정보를 찾기 위해 관련 테이블을 식별합니다.

2. United Airlines의 항공편 목록을 확인합니다.

3. Aberdeen으로 가는 항공편의 수를 세어봅니다.

SQL:

SELECT COUNT(\*)

FROM flights F

JOIN airlines A ON F.Airline = A.Airline

JOIN airports ADest ON F.DestAirport = ADest.AirportCode

WHERE A.Airline = 'United Airlines' AND ADest.City = 'Aberdeen';

1. **3-shot Chain-of-Thought Prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가요?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했나요?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있나요?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# airlines(uid, Airline, Abbreviation, Country);

# airports(City, AirportCode, AirportName, Country, CountryAbbrev);

# flights(Airline, FlightNo, SourceAirport, DestAirport);

Query: How many United Airlines flights go to City 'Aberdeen'?

단계별로:

1. United Airlines의 항공편 정보를 찾기 위해 관련 테이블을 식별합니다: `flights`와 `airlines`.

2. United Airlines의 항공편 목록을 찾습니다.

3. 목적지가 Aberdeen인 항공편의 수를 세어줍니다.

SQL:

SELECT COUNT(\*)

FROM flights F

JOIN airlines A ON F.Airline = A.Airline

JOIN airports ADest ON F.DestAirport = ADest.AirportCode

WHERE A.Airline = 'United Airlines' AND ADest.City = 'Aberdeen';

**[ 8 ]**

**Query:** Which template type code has most number of templates?

# Schema:

# Ref\_Template\_Types(Template\_Type\_Code, Template\_Type\_Description); # Templates(Template\_ID, Version\_Number, Template\_Type\_Code, Date\_Effective\_From, Date\_Effective\_To, Template\_Details);

# Documents(Document\_ID, Template\_ID, Document\_Name, Document\_Description, Other\_Details);

# Paragraphs(Paragraph\_ID, Document\_ID, Paragraph\_Text, Other\_Details);

1. **Zero-shot prompt**

Schema:

# Ref\_Template\_Types(Template\_Type\_Code, Template\_Type\_Description);

# Templates(Template\_ID, Version\_Number, Template\_Type\_Code, Date\_Effective\_From, Date\_Effective\_To, Template\_Details);

# Documents(Document\_ID, Template\_ID, Document\_Name, Document\_Description, Other\_Details);

# Paragraphs(Paragraph\_ID, Document\_ID, Paragraph\_Text, Other\_Details);

Query: Which template type code has most number of templates?

Explanation: Find the template type code that corresponds to the highest count of templates in the Templates table.

SQL:

SELECT Template\_Type\_Code, COUNT(\*) AS Num\_of\_Templates

FROM Templates

GROUP BY Template\_Type\_Code

ORDER BY Num\_of\_Templates DESC

LIMIT 1;

1. **3-shot prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가요?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했나요?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있나요?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# Ref\_Template\_Types(Template\_Type\_Code, Template\_Type\_Description);

# Templates(Template\_ID, Version\_Number, Template\_Type\_Code, Date\_Effective\_From, Date\_Effective\_To, Template\_Details);

# Documents(Document\_ID, Template\_ID, Document\_Name, Document\_Description, Other\_Details);

# Paragraphs(Paragraph\_ID, Document\_ID, Paragraph\_Text, Other\_Details);

Query: Which template type code has most number of templates?

단계별로:

1. 템플릿 유형 코드와 관련된 정보를 포함한 테이블을 식별합니다: `Ref\_Template\_Types`와 `Templates`.

2. 각 템플릿 유형 코드에 해당하는 템플릿 수를 세어봅니다.

3. 가장 많은 템플릿을 가진 템플릿 유형 코드를 식별합니다.

SQL:

SELECT Template\_Type\_Code, COUNT(\*) AS Num\_of\_Templates

FROM Templates

GROUP BY Template\_Type\_Code

ORDER BY Num\_of\_Templates DESC

LIMIT 1;

1. **Zero-shot Chain-of-Thought Prompt**

Schema:

# Ref\_Template\_Types(Template\_Type\_Code, Template\_Type\_Description);

# Templates(Template\_ID, Version\_Number, Template\_Type\_Code, Date\_Effective\_From, Date\_Effective\_To, Template\_Details);

# Documents(Document\_ID, Template\_ID, Document\_Name, Document\_Description, Other\_Details);

# Paragraphs(Paragraph\_ID, Document\_ID, Paragraph\_Text, Other\_Details);

Query: Which template type code has most number of templates?

단계별로:

1. 템플릿 유형 코드와 관련된 정보를 포함한 테이블을 식별합니다.

2. 각 템플릿 유형 코드에 해당하는 템플릿 수를 세어봅니다.

3. 가장 많은 템플릿을 가진 템플릿 유형 코드를 식별합니다.

SQL:

SELECT Template\_Type\_Code, COUNT(\*) AS Num\_of\_Templates

FROM Templates

GROUP BY Template\_Type\_Code

ORDER BY Num\_of\_Templates DESC

LIMIT 1;

1. **3-shot Chain-of-Thought Prompt**

예시 1:

Schema:

# customers(customer\_id, first\_name, last\_name, email);

# orders(order\_id, order\_date, customer\_id, amount);

Query: 고객들이 주문한 총 금액은 얼마인가요?

단계별로:

1. 주문 정보가 포함된 테이블을 식별합니다: `orders`.

2. `orders` 테이블에서 관련된 열을 찾습니다: `amount`.

3. 모든 주문의 `amount`를 합산합니다.

SQL: SELECT SUM(amount) FROM orders;

예시 2:

Schema:

# students(student\_id, first\_name, last\_name, grade);

# courses(course\_id, course\_name, teacher\_id);

# enrollments(student\_id, course\_id, enrollment\_date);

Query: 몇 명의 학생이 강좌에 등록했나요?

단계별로:

1. 등록 정보가 포함된 테이블을 식별합니다: `enrollments`.

2. `enrollments` 테이블에서 관련된 열을 찾습니다: `student\_id`.

3. 고유한 `student\_id` 값을 셉니다.

SQL: SELECT COUNT(DISTINCT student\_id) FROM enrollments;

예시 3:

Schema:

# employees(employee\_id, first\_name, last\_name, department\_id);

# departments(department\_id, department\_name, manager\_id);

Query: 각 부서에 몇 명의 직원이 있나요?

단계별로:

1. 직원 정보가 포함된 테이블을 식별합니다: `employees`.

2. `employees` 테이블에서 관련된 열을 찾습니다: `employee\_id` 및 `department\_id`.

3. `department\_id`로 데이터를 그룹화합니다.

4. 각 그룹의 `employee\_id` 수를 셉니다.

SQL: SELECT department\_id, COUNT(employee\_id) FROM employees GROUP BY department\_id;

Schema:

# Ref\_Template\_Types(Template\_Type\_Code, Template\_Type\_Description);

# Templates(Template\_ID, Version\_Number, Template\_Type\_Code, Date\_Effective\_From, Date\_Effective\_To, Template\_Details);

# Documents(Document\_ID, Template\_ID, Document\_Name, Document\_Description, Other\_Details);

# Paragraphs(Paragraph\_ID, Document\_ID, Paragraph\_Text, Other\_Details);

Query: Which template type code has most number of templates?

단계별로:

1. 템플릿 유형 코드와 관련된 정보를 포함한 테이블을 식별합니다.

2. 각 템플릿 유형 코드에 해당하는 템플릿 수를 세어봅니다.

3. 가장 많은 템플릿을 가진 템플릿 유형 코드를 식별합니다.

SQL:

SELECT Template\_Type\_Code, COUNT(\*) AS Num\_of\_Templates

FROM Templates

GROUP BY Template\_Type\_Code

ORDER BY Num\_of\_Templates DESC

LIMIT 1;