Describe the table (its meta-data, schema):

```
DESCRIBE employee;
```

## Altering the schema

```
ALTER TABLE employee ADD title char(20);
ALTER TABLE employee DROP title;
```

## Inserting values

```
INSERT INTO employee (name, salary, dept_id, cat_id)
values
(Smith, 2000.00, 2, 3);
```

## Select

SELECT *dept_id*
from *department*;

SELECT distinct *dept_id*
from *department*;

SELECT ∗
from *department*;

SELECT *name*, *salary* ∗ 1.1
from *employee*;

SELECT ∗
from *department*;

SELECT *name*, *salary* ∗ 1.1
from *employee*;

SELECT *name*, *salary* ∗ 1.1 as *new_salary*
from *employee*;

SELECT *name*, *salary*
from *employee*
where *salary* > 2000;

SELECT *name*, *salary*
from *employee*
where *salary* BETWEEN 2000 AND 3000;

SELECT *name*, *salary*
from *employee*
where *salary* NOT BETWEEN 2000 AND 3000;

SELECT *name*, *salary*
from *employee*
where *title* IN (‘*CEO*′, ‘*WebDeveloper*′);

```
SELECT name, salary
from employee
where title NOTIN ('CEO', 'WebDeveloper');
```

```
SELECT name, salary, dept_id
from employee
where employee.salary > 2000 and dept_id > 1;
```

---

```
SELECT name, salary
from employee
where name = "John";
```

```
SELECT name, salary
from employee
where name NOT LIKE "S%";
```

```
SELECT name, salary
from employee
where name LIKE "J%";
```

```
SELECT *
from employee
ORDER BY dept_id, salary;
```

## Group by

```
SELECT dept_id, avg(salary)
from employee
group by dept_id;
```

Appying a condition to a group of tuples by `having`:

```
SELECT dept_id, avg(salary)
from employee
group by dept_id;
having avg(salary) > 2000;
```

Notice that the above will take duplicates for averaging salary (salary dupes) which is esential for averaging.

Without any grouping

```
SELECT avg(salary)
from employee;
```