

Practice

1. 商品テーブルに、2つの商品を追加する

まず商品テーブルを確認する。

Query Editor

Query History

1

```
select * from items;
```

Data Output

Explain

Messages

Notifications

	id [PK] integer	category_id integer	name text	price integer
1		1	鯖	1000
2		2	鯛	800
3		4	鰹	700
4		5	牛	1200
5		7	豚	600
6		8	鳥	400
7		9	キャベツ	200
8		10	レタス	170
9		11	人参	80
10		12	大根	120

Query Editor

Query History

1

```
insert into items values (13, 1, '鰯', 150), (14, 2, '羊', 650);
```

2

```
select * from items;
```

Data Output

Explain

Messages

Notifications

	id [PK] integer	category_id integer	name text	price integer
1		1	鯖	1000
2		2	鯛	800
3		4	鰹	700
4		5	牛	1200
5		7	豚	600
6		8	鳥	400
7		9	キャベツ	200
8		10	レタス	170
9		11	人参	80
10		12	大根	120
11		13	鰯	150
12		14	羊	650

2. 商品テーブルの全商品の価格を、1 割引きに変更する

```

1 update items set price = price * 0.9;
2 select * from items;

```

Data Output Explain Messages Notifications

	id [PK] integer	category_id integer	name text	price integer
1	1	1	鯖	900
2	2	2	鯛	720
3	4	1	鰹	630
4	5	2	牛	1080
5	7	2	豚	540
6	8	2	鳥	360
7	9	3	キャベツ	180
8	10	3	レタス	153
9	11	3	人参	72
10	12	3	大根	108
11	13	1	鰯	135
12	14	2	羊	585

3. 従業員テーブルから、2013 年 3 月 31 日以前に退社した人を削除する

最初に従業員テーブルを確認する。

```

1 select * from employees limit 5;

```

Data Output Explain Messages Notifications

	id [PK] integer	name text	email text	start_date date	end_date date
1	1	ツヤコ	tsuyako@example.com	2000-04-02	[null]
2	2	チュウジ	choji@example.com	2000-06-03	2003-09-30
3	3	カツノリ	katsunori@example.com	2000-08-02	[null]
4	4	ランポ	ranpo@example.com	2001-05-01	2005-06-30
5	5	リョウスケ	ryousuke@example.com	2003-10-01	[null]

```

1 delete from employees where end_date <= '2013-3-31';
2 select * from employees limit 5;

```

Data Output Explain Messages Notifications

	id [PK] integer	name text	email text	start_date date	end_date date
1	1	ツヤコ	tsuyako@example.com	2000-04-02	[null]
2	3	カツノリ	katsunori@example.com	2000-08-02	[null]
3	5	リョウスケ	ryousuke@example.com	2003-10-01	[null]
4	6	ユリコ	yuriko@example.com	2005-09-01	[null]
5	7	ノリカ	norika@example.com	2005-11-01	2014-03-31

4. 商品テーブルの ID について、歯抜けになっている数値のうち、最も小さい数値を取得する

まず商品テーブルを確認する。

Query Editor Query History

1 `select * from items limit 5;`

Data Output Explain Messages Notifications

	id [PK] integer	category_id integer	name text	price integer
1		1	鯖	900
2		2	鯛	720
3		4	鯉	630
4		5	牛	1080
5		7	豚	540

歯抜けしている最小値は、3であることが分かる。

generate_series() : Pythonにおけるrange()やnp.arange()と同じ

Query Editor Query History

1 `select generate_series(1, (SELECT max(id) FROM items)) limit 5;`

Data Output Explain Messages Notifications

	generate_series integer
1	1
2	2
3	3
4	4
5	5

上記のようなIDの連番テーブルを作成し、商品テーブルと結合させ、NULLとなったところを最小値として取り出す。

Query Editor Query History

1 `select min(B) as absence`
2 `from items as A right outer join generate_series(1, (SELECT max(id) FROM items)) as B`
3 `on A.id = B`
4 `where A.id is null;`

Data Output Explain Messages Notifications

	absence integer
1	3

5. 地方名と都道府県名を、都道府県ごとに出力する

地方テーブルを確認する。

Query Editor Query History

```
1 select * from areas;
```

Data Output Explain Messages Notifications

	code [PK] character varying (2)	name text
1	00	全国
2	10	北海道
3	20	東北
4	30	関東
5	40	中部
6	50	近畿
7	60	中国
8	70	四国
9	80	九州

都道府県テーブルを確認する。

Query Editor Query History

```
1 select * from prefectures limit 10;
```

Data Output Explain Messages Notifications

	code [PK] character varying (2)	area_code character varying (2)	name text	acreage double precision
1	01	10	北海道	83456.87
2	02	20	青森県	9644.54
3	03	20	岩手県	15278.89
4	04	20	宮城県	7285.76
5	05	20	秋田県	11636.25
6	06	20	山形県	9323.46
7	07	20	福島県	13782.76
8	08	30	茨城県	6095.72
9	09	30	栃木県	6408.28
10	10	30	群馬県	6362.33

INNER JOINを行う。

```

1 select a.name, p.name
2 from areas as a inner join prefectures as p
3 on a.code = p.area_code
4 limit 10;

```

Data Output Explain Messages Notifications

	name text	name text
1	北海道	北海道
2	東北	青森県
3	東北	岩手県
4	東北	宮城県
5	東北	秋田県
6	東北	山形県
7	東北	福島県
8	関東	茨城県
9	関東	栃木県
10	関東	群馬県

6. 総人口を求める

人口テーブルを確認する。

```

1 select * from populations limit 10;

```

Data Output Explain Messages Notifications

	id [PK] integer	prefecture_code character (2)	age_range_code character (1)	sex_code character (1)	population integer
1	1	01	1	m	335353
2	2	02	1	m	87585
3	3	03	1	m	86612
4	4	04	1	m	158024
5	5	05	1	m	63335
6	6	06	1	m	76678
7	7	07	1	m	141275
8	8	08	1	m	205036
9	9	09	1	m	138441
10	10	10	1	m	140983

集約関数を使用すれば良い。

```

1 select sum(population) as total_population from populations;

```

Data Output Explain Messages Notifications

	total_population bigint
1	127080929

7. 九州地方の各県で、男女別の人口を求める

都道府県テーブル（pr）を確認する。

Query Editor Query History

1 `select * from prefectures limit 10;`

Data Output Explain Messages Notifications

	code [PK] character varying (2)	area_code character varying (2)	name text	acreage double precision
1	01	10	北海道	83456.87
2	02	20	青森県	9644.54
3	03	20	岩手県	15278.89
4	04	20	宮城県	7285.76
5	05	20	秋田県	11636.25
6	06	20	山形県	9323.46
7	07	20	福島県	13782.76
8	08	30	茨城県	6095.72
9	09	30	栃木県	6408.28
10	10	30	群馬県	6362.33

性別（s）テーブルを確認する。

Query Editor Query History

1 `select * from sexes;`

Data Output Explain Messages Notifications

	code [PK] character (1)	name character
1	f	女
2	m	男

地方テーブル（a）を確認する。

Query Editor Query History

1 `select * from areas;`

Data Output Explain Messages Notifications

	code [PK] character varying (2)	name text
1	00	全国
2	10	北海道
3	20	東北
4	30	関東
5	40	中部
6	50	近畿
7	60	中国
8	70	四国
9	80	九州

人口テーブル（po）を確認する。

Query Editor Query History

1 `select * from populations limit 10;`

Data Output Explain Messages Notifications

	id [PK] integer	prefecture_code character (2)	age_range_code character (1)	sex_code character (1)	population integer
1	1	01	1	m	335353
2	2	02	1	m	87585
3	3	03	1	m	86612
4	4	04	1	m	158024
5	5	05	1	m	63335
6	6	06	1	m	76678
7	7	07	1	m	141275
8	8	08	1	m	205036
9	9	09	1	m	138441
10	10	10	1	m	140983

必要なテーブルを結合したものについて、全体構造は次のようになる。

Query Editor Query History

1 `select pr.code, s.code, a.code, a.name, pr.name as prefecture, s.name as sex, sum(po.population) as population`
2
3 `from populations as po inner join prefectures as pr`
4 `on po.prefecture_code = pr.code`
5
6 `inner join sexes as s`
7 `on po.sex_code = s.code`
8
9 `inner join areas as a`
10 `on pr.area_code = a.code`
11
12 `group by pr.code, s.code, a.code`
13 `having a.name = '九州'`
14 `order by pr.code asc, s.name desc`
15
16 `limit 5;`

Data Output Explain Messages Notifications

	code character varying (2)	code character (1)	code character varying (2)	name text	prefecture text	sex character varying (1)	population bigint
1	40	m	80	九州	福岡県	男	2373377
2	40	f	80	九州	福岡県	女	2662055
3	41	m	80	九州	佐賀県	男	398293
4	41	f	80	九州	佐賀県	女	448456
5	42	m	80	九州	長崎県	男	662196

pr.codeでグループ化して2つに分け、さらにその中でs.codeでグループ化して2つに分ける。
a.codeは80という1つの値しかないので、全ての値が80となっている。

尚、a.codeも入れた理由としては、havingを使用する際に、group byの中でareaテーブルを使用する必要があったためである。
havingは、group byで使用した列名にしか適用することができない。
そのため今回の場合であれば、group byに入れるものは、a.codeでなくa.nameでも問題ない。

```

1  select pr.name as prefecture, s.name as sex, sum(po.population) as population
2
3  from populations as po inner join prefectures as pr
4  on po.prefecture_code = pr.code
5
6  inner join sexes as s
7  on po.sex_code = s.code
8
9  inner join areas as a
10 on pr.area_code = a.code
11
12 group by pr.code, s.code, a.code
13 having a.name = '九州'
14 order by pr.code asc, s.name desc
15
16 limit 5;

```

Data Output Explain Messages Notifications

	prefecture text	sex character varying (1)	population bigint
1	福岡県	男	2373377
2	福岡県	女	2662055
3	佐賀県	男	398293
4	佐賀県	女	448456
5	長崎県	男	662196

8. 地方ごとに各世代の男女の人口を求める

まず、必要なテーブルを結合して、a.codeでグループ化する。

```

1  select
2  a.code as area_code,
3  a.name as area
4
5  from populations as po
6  inner join prefectures as pr
7  on po.prefecture_code = pr.code
8
9  inner join areas as a
10 on pr.area_code = a.code
11
12 group by a.code
13 order by a.code asc
14 limit 5;

```

Data Output Explain Messages Notifications

	area_code character varying (2)	area text
1	10	北海道
2	20	東北
3	30	関東
4	40	中部
5	50	近畿

その後、グループ化したものに対して、集約関数「SUM」を使用する。
SUMの中でCASEを用いて、各条件を満たすデータを取り出す。

CASE文：CASE WHEN（条件）THEN（条件を満たしたら表示する内容）ELSE（条件を満たしていなかったら表示する内容）END


```
1 select
2 a.name as area,
3 sum(case when po.sex_code = 'm' and po.age_range_code = '1' then po.population else 0 end) as m_below_15,
4 sum(case when po.sex_code = 'm' and po.age_range_code = '2' then po.population else 0 end) as m_from_15_to_64,
5 sum(case when po.sex_code = 'm' and po.age_range_code = '3' then po.population else 0 end) as m_over65,
6 sum(case when po.sex_code = 'f' and po.age_range_code = '1' then po.population else 0 end) as f_below_15,
7 sum(case when po.sex_code = 'f' and po.age_range_code = '2' then po.population else 0 end) as f_from_15_to_64,
8 sum(case when po.sex_code = 'f' and po.age_range_code = '3' then po.population else 0 end) as f_over_65
9
10 from populations as po
11 inner join prefectures as pr
12 on po.prefecture_code = pr.code
13
14 inner join areas as a
15 on pr.area_code = a.code
16
17 group by a.code
18 order by a.code asc
19 limit 3;
```

Data Output Explain Messages Notifications

	area text	m_below_15 bigint	m_from_15_to_64 bigint	m_over65 bigint	f_below_15 bigint	f_from_15_to_64 bigint	f_over_65 bigint	
1	北海道	335353	1695591	567141	321959	1786578	790927	
2	東北	613509	2859971	973967	585227	2851235	1406716	
3	関東	2745759	14357062	3896349	2617355	13664771	4924146	

今回の場合、「SUM」を使用する必要はなさそうだが、グループ化をしているため、集約関数を使わなければならない。