

## Массивы

Создадим новую таблицу pilots

```
test=# CREATE TABLE pilots
test=# (
test=# pilot_name text,
test=# schedule integer[]
test=# );
CREATE TABLE
```

```
test=# INSERT INTO pilots
```

```
test=# VALUES ( 'Ivan', '{ 1, 3, 5, 6, 7 }'::integer[] ),
```

```
test=# ( 'Petr', '{ 1, 2, 5, 7 }'::integer[] ),
```

```
test=# ( 'Pavel', '{ 2, 5 }'::integer[] ),
```

```
test=# ( 'Boris', '{ 3, 5, 6 }'::integer[] );
```

```
INSERT 0 4
```

Заполним ее

данными с

ПОМОЩЬЮ МАССИВОВ

Посмотрим, что к чему там. Выглядит здорово

```
test=# SELECT * FROM pilots;
pilot_name | schedule
```

pilot_name	schedule
Ivan	{1,3,5,6,7}
Petr	{1,2,5,7}
Pavel	{2,5}
Boris	{3,5,6,7}

(4 строки)

Самую малость update(ов)

```
test=# UPDATE pilots
```

```
test=# SET schedule = schedule || 7
```

```
test=# WHERE pilot_name = 'Boris';
```

```
UPDATE 1
```

```
test=# UPDATE pilots
```

```
test=# SET schedule = array_append( schedule, 6 )
```

```
test=# WHERE pilot_name = 'Pavel';
```

```
UPDATE 1
```

```
test=# UPDATE pilots
```

```
test=# SET schedule = array_prepend( 1, schedule )
```

```
test=# WHERE pilot_name = 'Pavel';
```

```
UPDATE 1
```

```
test=# UPDATE pilots
```

```
test=# SET schedule = array_remove( schedule, 5 )
```

```
test=# WHERE pilot_name = 'Ivan';
```

```
UPDATE 1
```

```
test=# UPDATE pilots
```

```
test=# SET schedule[ 1 ] = 2, schedule[ 2 ] = 3
```

```
test=# WHERE pilot_name = 'Petr';
```

```
UPDATE 1
```

```
test=# UPDATE pilots
```

```
test=# SET schedule[ 1:2 ] = ARRAY[ 2, 3 ]
```

```
test=# WHERE pilot_name = 'Petr';
```

```
UPDATE 1
```

Проверим что к чему, все поменялось,  
отлично

```
test=# SELECT * FROM pilots;
 pilot_name |  schedule
-----+-----
 Boris      | {3,5,6,7,7}
 Pavel      | {1,2,5,6}
 Ivan       | {1,3,6,7}
 Petr       | {2,3,5,7}
(4 строки)
```

```
test=# SELECT * FROM pilots
test=# WHERE array_position( schedule, 3 ) IS NOT NULL;
 pilot_name |  schedule
-----+-----
 Boris      | {3,5,6,7,7}
 Ivan       | {1,3,6,7}
 Petr       | {2,3,5,7}
(3 строки)
```

Как видим, нулевой  
элемент массива  
нам не показало

Теперь те, у кого есть 1 и 7

```
test=# SELECT * FROM pilots
test=# WHERE schedule @> '{ 1, 7 } '::integer[];
 pilot_name |  schedule
-----+-----
 Ivan       | {1,3,6,7}
(1 строка)
```

```
test=# SELECT * FROM pilots
test=# WHERE schedule && ARRAY[ 2, 5 ];
 pilot_name |  schedule
-----+-----
 Boris      | {3,5,6,7,7}
 Pavel      | {1,2,5,6}
 Petr       | {2,3,5,7}
(3 строки)
```

Те, у кого есть 2 и/или 5

Теперь те, у кого нет 2  
и/или 5

```
test=# SELECT * FROM pilots
test=# WHERE NOT ( schedule && ARRAY[ 2, 5 ] );
 pilot_name |  schedule
-----+-----
 Ivan       | {1,3,6,7}
(1 строка)
```

```
test=# SELECT unnest( schedule ) AS days_of_week
test=# FROM pilots
test=# WHERE pilot_name = 'Ivan';
 days_of_week
-----
 1
 3
 6
 7
(4 строки)
```

Дни у Ивана

## JSON

Для тестов создадим новую таблицу

```
test=# CREATE TABLE pilot_hobbies
test=# (
test(# pilot_name text,
test(# hobbies jsonb
test(# );
CREATE TABLE
```

```
test=# INSERT INTO pilot_hobbies
test=# VALUES ( 'Ivan',
test(# '{ "sports": [ "футбол", "плавание" ],
test'# "home_lib": true, "trips": 3
test'# }':::jsonb
test(# ),
test=# ( 'Petr',
test(# '{ "sports": [ "теннис", "плавание" ],
test'# "home_lib": true, "trips": 2
test'# }':::jsonb
test(# ),
test=# ( 'Pavel',
test(# '{ "sports": [ "плавание" ],
test'# "home_lib": false, "trips": 4
test'# }':::jsonb
test(# ),
test=# ( 'Boris',
test(# '{ "sports": [ "футбол", "плавание", "теннис" ],
test'# "home_lib": true, "trips": 0
test'# }':::jsonb
test(# );
INSERT 0 4
```

И заполним ее  
значениями в формате  
JSON

Выглядит интересно

```
test=# SELECT * FROM pilot_hobbies;
 pilot_name |                               hobbies
-----+-----
 Ivan      | {"trips": 3, "sports": ["футбол", "плавание"], "home_lib": true}
 Petr      | {"trips": 2, "sports": ["теннис", "плавание"], "home_lib": true}
 Pavel     | {"trips": 4, "sports": ["плавание"], "home_lib": false}
 Boris     | {"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
(4 строки)
```

Выбираем футболистов так

```
test=# SELECT * FROM pilot_hobbies
test=# WHERE hobbies @> '{ "sports": [ "футбол" ] }':::jsonb;
 pilot_name |                               hobbies
-----+-----
 Ivan      | {"trips": 3, "sports": ["футбол", "плавание"], "home_lib": true}
 Boris     | {"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
(2 строки)
```

Либо же вот так

```
test=# SELECT pilot_name, hobbies->'sports' AS sports
test=# FROM pilot_hobbies
test=# WHERE hobbies->'sports' @> '[ "футбол" ] '::jsonb;
pilot_name | sports
-----+-----
Ivan       | ["футбол", "плавание"]
Boris      | ["футбол", "плавание", "теннис"]
(2 строки)
```

```
test=# SELECT count( * )
test=# FROM pilot_hobbies
test=# WHERE hobbies ? 'sport';
count
-----
0
(1 строка)
```

Проверка на ключ, ключа sport у нас нет

А вот sports есть, работает!

```
test=# SELECT count( * )
test=# FROM pilot_hobbies
test=# WHERE hobbies ? 'sports';
count
-----
4
(1 строка)
```

Пажиллой update

```
test=# UPDATE pilot_hobbies
test=# SET hobbies = hobbies || '{ "sports": [ "хоккей" ] }'
test=# WHERE pilot_name = 'Boris';
UPDATE 1
```

ВЫГЛЯДИТ ЭТО ВОТ ТАК

```
test=# SELECT pilot_name, hobbies
test=# FROM pilot_hobbies
test=# WHERE pilot_name = 'Boris';
pilot_name | hobbies
-----+-----
Boris      | {"trips": 0, "sports": ["хоккей"], "home_lib": true}
(1 строка)
```

Делаем из Бориса спортсмена

```
test=# UPDATE pilot_hobbies
test=# SET hobbies = jsonb_set( hobbies, '{ sports, 1 }', '"футбол"' )
test=# WHERE pilot_name = 'Boris';
UPDATE 1
```

## И правда спортсмен

```
test=# SELECT pilot_name, hobbies
test=# FROM pilot_hobbies
test=# WHERE pilot_name = 'Boris';
```

pilot_name	hobbies
Boris	{"trips": 0, "sports": ["хоккей", "футбол"], "home_lib": true}

(1 строка)

## Контрольные вопросы и задания

### Задание 1

Создаем новую табличку

```
test=# CREATE TABLE test_numeric
test=# ( measurement numeric(5, 2),
test=# description text
test=# );
CREATE TABLE
```

При попытке ввода данных получаем следующие ошибки

```
test=# INSERT INTO test_numeric
test=# VALUES ( 999.9999, 'Какое-то измерение ' );
ОШИБКА: переполнение поля numeric
ПОДРОБНОСТИ: Поле с точностью 5, порядком 2 должно округляться до абсолютного значения меньше чем 10^3.
test=# INSERT INTO test_numeric
test=# VALUES ( 999.90009, 'Еще одно измерение' );
INSERT 0 1
test=# INSERT INTO test_numeric
test=# VALUES ( 999.1111, 'И еще измерение' );
INSERT 0 1
test=# INSERT INTO test_numeric
test=# VALUES ( 998.9999, 'И еще одно' );
INSERT 0 1
```

### Задание 2

Удаляем старую таблицу

```
test=# DROP TABLE test_numeric;
DROP TABLE
```

```
test=# CREATE TABLE test_numeric
test=# ( measurement numeric,
test=# description text
test=# );
CREATE TABLE
```

Создаем новую чуть иначе

Заполняем данными

```
test=# INSERT INTO test_numeric
test=# VALUES ( 1234567890.0987654321,
test=# 'Точность 20 знаков, масштаб 10 знаков' );
INSERT 0 1
test=# INSERT INTO test_numeric
test=# VALUES ( 1.5,
test=# 'Точность 2 знака, масштаб 1 знак' );
INSERT 0 1
test=# INSERT INTO test_numeric
test=# VALUES ( 0.12345678901234567890,
test=# 'Точность 21 знак, масштаб 20 знаков' );
INSERT 0 1
test=# INSERT INTO test_numeric
test=# VALUES ( 1234567890,
test=# 'Точность 10 знаков, масштаб 0 знаков (целое число)' );
INSERT 0 1
```

```
test=# select * from test_numeric
test=# ;
```

measurement	description
1234567890.0987654321	Точность 20 знаков, масштаб 10 знаков
1.5	Точность 2 знака, масштаб 1 знак
0.12345678901234567890	Точность 21 знак, масштаб 20 знаков
1234567890	Точность 10 знаков, масштаб 0 знаков (целое число)

(4 строки)

Смотрим данные

### Задание 3

```
test=# SELECT 'NaN'::numeric > 10000;
?column?
-----
t
(1 строка)
```

### Задание 4

Почему так?

```
test=# SELECT '5e-324'::double precision > '4e-324'::double precision;
?column?
-----
f
(1 строка)
```

А числа то одинаковые, получается

```
test=# SELECT '5e-324'::double precision;
float8
-----
5e-324
(1 строка)
```

```
test=# SELECT '4e-324'::double precision;
float8
-----
5e-324
(1 строка)
```

Самостоятельно

```
test=# SELECT '3e-324'::double precision='4e-324'::double precision;
?column?
-----
t
(1 строка)
```

### Задание 5

```
test=# SELECT 'Inf'::double precision > 1E+308;
?column?
-----
t
(1 строка)
```

Самостоятельно

```
test=# SELECT '-Inf'::double precision < '3e-324'::double precision;
?column?
-----
t
(1 строка)
```

### Задание 6

NaN

```
test=# SELECT 0.0 * 'Inf'::real;
?column?
-----
NaN
(1 строка)
```

TRUE NaN

```
test=# select 'NaN'::real > 'Inf'::real
?column?
-----
t
(1 строка)
```

## Задание 7

```
test=# CREATE TABLE test_serial
test=# ( id serial,
test=# name text
test=# );
CREATE TABLE
test=# INSERT INTO test_serial ( name ) VALUES ( 'Вишневая' );
INSERT 0 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );
INSERT 0 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Зеленая' );
INSERT 0 1
-----
test=# INSERT INTO test_serial ( id, name ) VALUES ( 10, 'Прохладная' );
INSERT 0 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Луговая' );
INSERT 0 1
```

## Задание 8

```
test=# CREATE TABLE test_serial
test=# ( id serial PRIMARY KEY,
test=# name text
test=# );
CREATE TABLE
test=# INSERT INTO test_serial ( name ) VALUES ( 'Вишневая' );
INSERT 0 1
test=# INSERT INTO test_serial ( id, name ) VALUES ( 2, 'Прохладная' );
INSERT 0 1

test=# INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );
ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "test_serial_pkey"
ПОДРОБНОСТИ: Ключ "(id)=(2)" уже существует.
test=# INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );
INSERT 0 1
```

Ошибка, потому что запись с таким id уже есть

Четверки нет ((((((((((

```
test=# INSERT INTO test_serial ( name ) VALUES ( 'Зеленая' );
INSERT 0 1
test=# DELETE FROM test_serial WHERE id = 4;
DELETE 1
test=# INSERT INTO test_serial ( name ) VALUES ( 'Луговая' );
INSERT 0 1
test=# SELECT * FROM test_serial;
 id |      name
-----+-----
  1 | Вишневая
  2 | Прохладная
  3 | Грушевая
  5 | Луговая
(4 строки)
```

## Задание 9

Григорианский

## Задание 10

Из-за ограничений по памяти. Эти данные хранятся в восьми байтах.

## Задание 11

```
test=# SELECT current_time; test=# SELECT current_time::time( 0 );
      current_time              current_time
-----
19:01:43.854231+03             19:01:47
(1 строка)                    (1 строка)
```

```
test=# SELECT current_time::time( 3 );
      current_time
-----
19:01:51.575
(1 строка)
```

```
test=# SELECT '2004-10-19 10:23:54.421214'::timestamp(2);
      timestamp
-----
2004-10-19 10:23:54.42
(1 строка)
```

```
test=# SELECT '2004-10-19 10:23:54.421214'::timestamp(0);
      timestamp
-----
2004-10-19 10:23:54
(1 строка)
```

```
test=# SELECT 'P0001-02-03T04:05:06.321321'::interval(1);
      interval
-----
1 year 2 mons 3 days 04:05:06.3
(1 строка)
```

Ответ: Потому что дни целочисленный формат.

## Задание 12

```
test=# SET datestyle TO
test=# 'German, DMY';
SET
test=# SELECT
test=# '18-05-2016'
test=# ::timestamp;
      timestamp
-----
18.05.2016 00:00:00
(1 строка)
```



### Задание 13-14 is sadge

### Задание 15

```
test=# SELECT to_char( current_timestamp, 'hh:mi:ss dd-mm-yyyy');
         to_char
```

```
-----
```

```
07:13:41 26-10-2021
```

```
(1 строка)
```

### Задание 16

```
test=# SELECT 'Feb 29, 2015'::date;
```

```
ОШИБКА: значение поля типа date/time вне диапазона: "Feb 29, 2015"
```

```
СТРОКА 1: SELECT 'Feb 29, 2015'::date;
```

### Задание 17

```
test=# SELECT '21:15:16:22'::time;
```

```
ОШИБКА: неверный синтаксис для типа time: "21:15:16:22"
```

```
СТРОКА 1: SELECT '21:15:16:22'::time;
```

### Задание 18

15!

```
test=# SELECT ( '2016-09-16'::date - '2016-09-01'::date );
         ?column?
```

```
-----
```

```
15
```

```
(1 строка)
```

### Задание 19

Если вместо «-» написать «+», то пишет ошибку. Так как мы плюсуем два времени суток, то ничего и не получается. Вместо другого времени нужно брать тип интервала.

### Задание 20

Будет выведена текущая дата + 1 месяц.

Если убрать псевдоним, то в СУБД PostgreSQL будет выведено «?column?»

## Задание 21

Он выводит последний день месяца. В случае февраля это 29 число.

```
test=# SELECT ('2016-01-31'::date + '1 mon'::interval ) AS new_date;
          new_date
-----
Mon 29 Feb 00:00:00 2016
(1 строка)
```

Он выводит тоже 29 число, здесь это получается из-за того, что здесь он уже прибавляет полноценный месяц.

```
test=# SELECT ('2016-02-29'::date + '1 mon'::interval ) AS new_date;
          new_date
-----
Tue 29 Mar 00:00:00 2016
(1 строка)
```

## Задание 22

```
test=# SET intervalstyle TO 'postgres_verbose';
SET
test=# SELECT ('1 mon '::interval) AS new_date;
          new_date
-----
@ 1 mon
(1 строка)
```

## Задание 23

```
test=# SELECT ('2016-09-16'::date - '2015-09-01'::date );SELECT ('2016-09-16'::timestamp - '2015-09-01'::timestamp);
?column?
-----
381
(1 строка)

?column?
-----
@ 381 days
(1 строка)
```

date и timestamp почти одно и тоже. Поэтому результаты одни и те же.

## Задание 24

Чтобы работало делаем так:

```
test=# SELECT ('20:34:35'::time - '1 hour'::interval);
?column?
-----
19:34:35
(1 строка)

test=# SELECT ('2016-09-16'::date - 1 );
?column?
-----
15-09-2016
(1 строка)
```

Из времени нельзя вычесть 1

## Задание 25

1

SELECT ( date\_trunc( 'mon',

2

timestamp '1999-11-27 12:34:56.987654' ) );

Результат

План выполн...

Сообщения

Notifications

date\_trunc

timestamp without time zone

1

1999-11-01 00:00:00

## Задание 26

1

SELECT ( date\_trunc( 'min',

2

interval '1 hour 36 seconds' ) );

Результат

План выполн...

Сообщения

Not

date\_trunc

interval

1

01:00:00

## Задание 27

1

SELECT extract(

2

'year' from timestamp '1999-11-27 12:34:56.123459'

3

);

Результат

План выполн...

Сообщения

Notifications

date\_part

double precision

1

1999

✓

Запрос

Задание 28

Query Editor История запр...

1

SELECT extract(

2

'year' from interval '30 months'

3

);

Результат

План выполн...

Сообщения

No

	date_part	
	double precision	
1		2

Задание 29

Равнозначны. Т.к. непустая строка равна ИСТИНЕ.

А последний запрос не работает, ибо почему-то нельзя сравнивать boolean и int.

1

SELECT \* FROM databases WHERE is\_open\_source <> 'TRUE';

2

3

4

5

6

7

Результат

План выполн...

Сообщения

Notifications

	is_open_source	dbms_name	
	boolean	text	
1	false	Oracle	
2	false	MS SQL Server	

Задание 30

Не работает 2, 6 и 8 запрос, ибо для str нужно добавлять апострофы, а int нельзя без явного преобразования переместить в boolean.

Задание 31

1

SELECT age(birthday) FROM birthdays

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

## Задание 32

```
1 SELECT array_length(array[1,2,3], 1)
2
```

Data Output	Explain	Messages	Notifications				
<table><tr><th>array_length</th><th>integer</th></tr><tr><td>1</td><td>3</td></tr></table>	array_length	integer	1	3			
array_length	integer						
1	3						

## Задание 33

```
1 SELECT meal[1][1] FROM pilots1
```

Data Output	Explain	Messages	Notifications										
<table><tr><th>meal</th><th>text</th></tr><tr><td>1</td><td>сосиска</td></tr><tr><td>2</td><td>сосиска</td></tr><tr><td>3</td><td>сосиска</td></tr><tr><td>4</td><td>сосиска</td></tr></table>	meal	text	1	сосиска	2	сосиска	3	сосиска	4	сосиска			
meal	text												
1	сосиска												
2	сосиска												
3	сосиска												
4	сосиска												

## Задание 34

```
UPDATE pilot_hobbies
SET hobbies = jsonb_set( hobbies, '{ home_lib }', 'false')
WHERE pilot_name = 'Pavel';
```

```
1 SELECT pilot_name, hobbies->'home_lib' FROM pilot_hobbies;
```

Data Output	Explain	Messages	Notifications												
<table><tr><th>pilot_name</th><th>?column?</th></tr><tr><td>text</td><td>jsonb</td></tr><tr><td>1 Ivan</td><td>true</td></tr><tr><td>2 Petr</td><td>true</td></tr><tr><td>3 Boris</td><td>true</td></tr><tr><td>4 Pavel</td><td>false</td></tr></table>	pilot_name	?column?	text	jsonb	1 Ivan	true	2 Petr	true	3 Boris	true	4 Pavel	false			
pilot_name	?column?														
text	jsonb														
1 Ivan	true														
2 Petr	true														
3 Boris	true														
4 Pavel	false														

Задание 35

1 SELECT '{ "sports": "хоккей"}'::jsonb || '{"trips": 5}'::jsonb->'trips'

Data Output Explain Messages Notifications

	?column? jsonb
1	5

Задание 36

UPDATE pilot\_hobbies  
SET hobbies= hobbies || '{"chips": 5}'::jsonb  
WHERE pilot\_name='Pavel'

1 SELECT \* FROM pilot\_hobbies

Data Output Explain Messages Notifications

	pilot_name text	hobbies jsonb
1	Ivan	{ "trips": 3, "sports": ["футбол", "плавание"], "home_lib": true }
2	Petr	{ "trips": 2, "sports": ["теннис", "плавание"], "home_lib": true }
3	Boris	{ "trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true }
4	Pavel	{ "chips": 5, "trips": 4, "sports": ["плавание"], "home_lib": false }

Задание 37

UPDATE pilot\_hobbies  
SET hobbies= hobbies - 'chips'  
WHERE pilot\_name='Pavel'

1 SELECT \* FROM pilot\_hobbies  
2  
3

Data Output Explain Messages Notifications

	pilot_name text	hobbies jsonb
1	Ivan	{ "trips": 3, "sports": ["футбол", "плавание"], "home_lib": true }
2	Petr	{ "trips": 2, "sports": ["теннис", "плавание"], "home_lib": true }
3	Boris	{ "trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true }
4	Pavel	{ "trips": 4, "sports": ["плавание"], "home_lib": false }