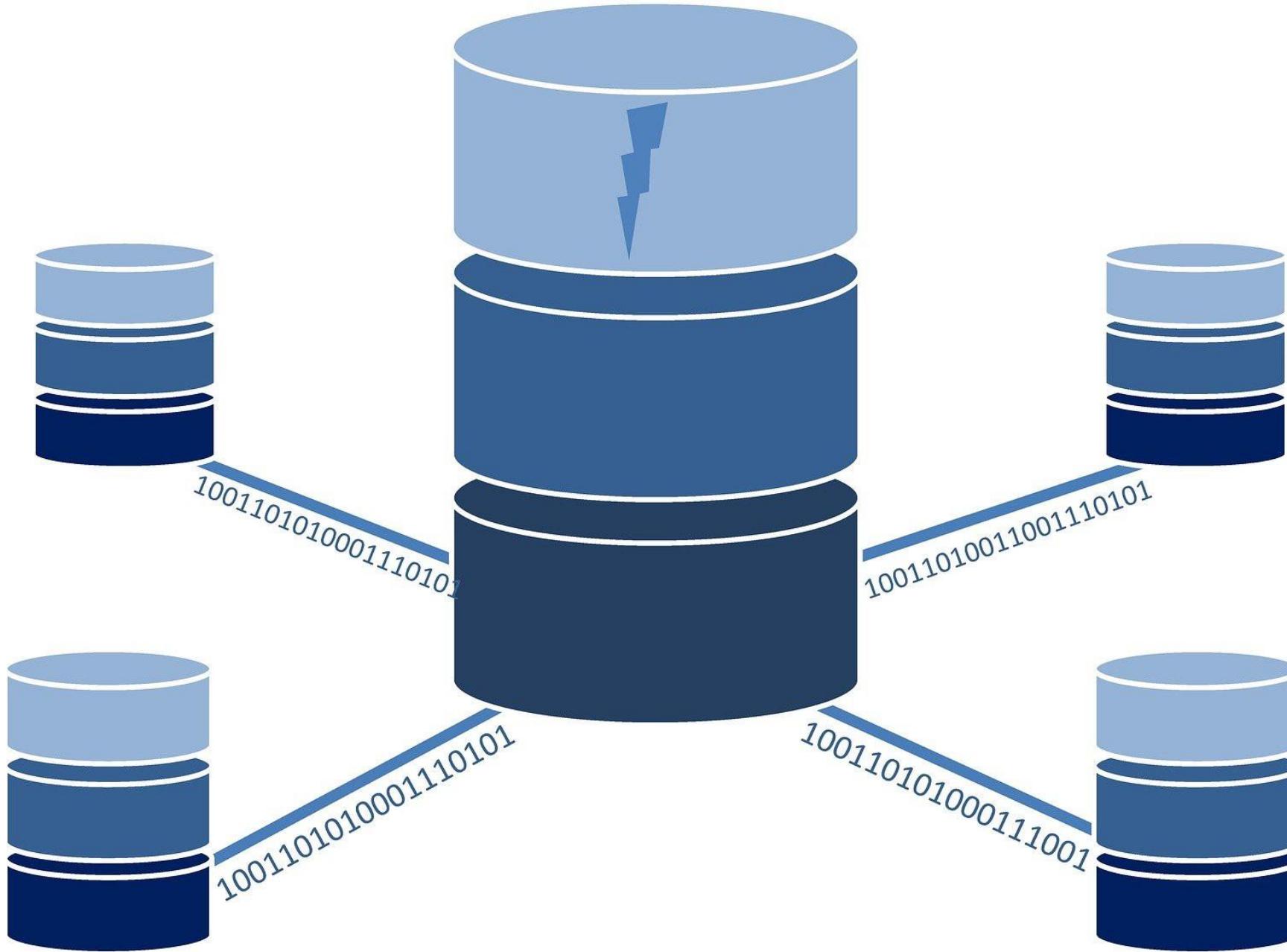
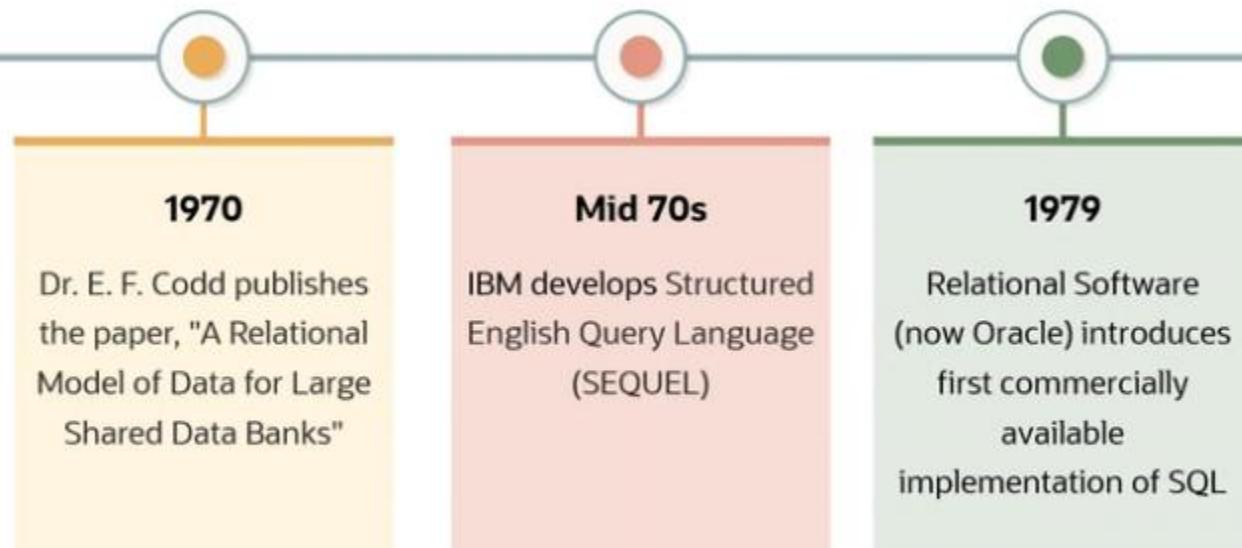


[Sql syllabus \(slideshare.net\)](#)



Brief History of SQL



SQL: Features

Structured Query Language (SQL) is:

The ANSI standard language for operating relational databases

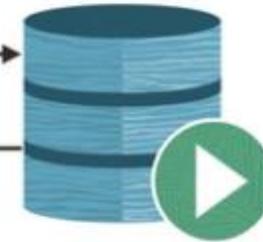
Efficient, easy to learn and use

Functionally complete

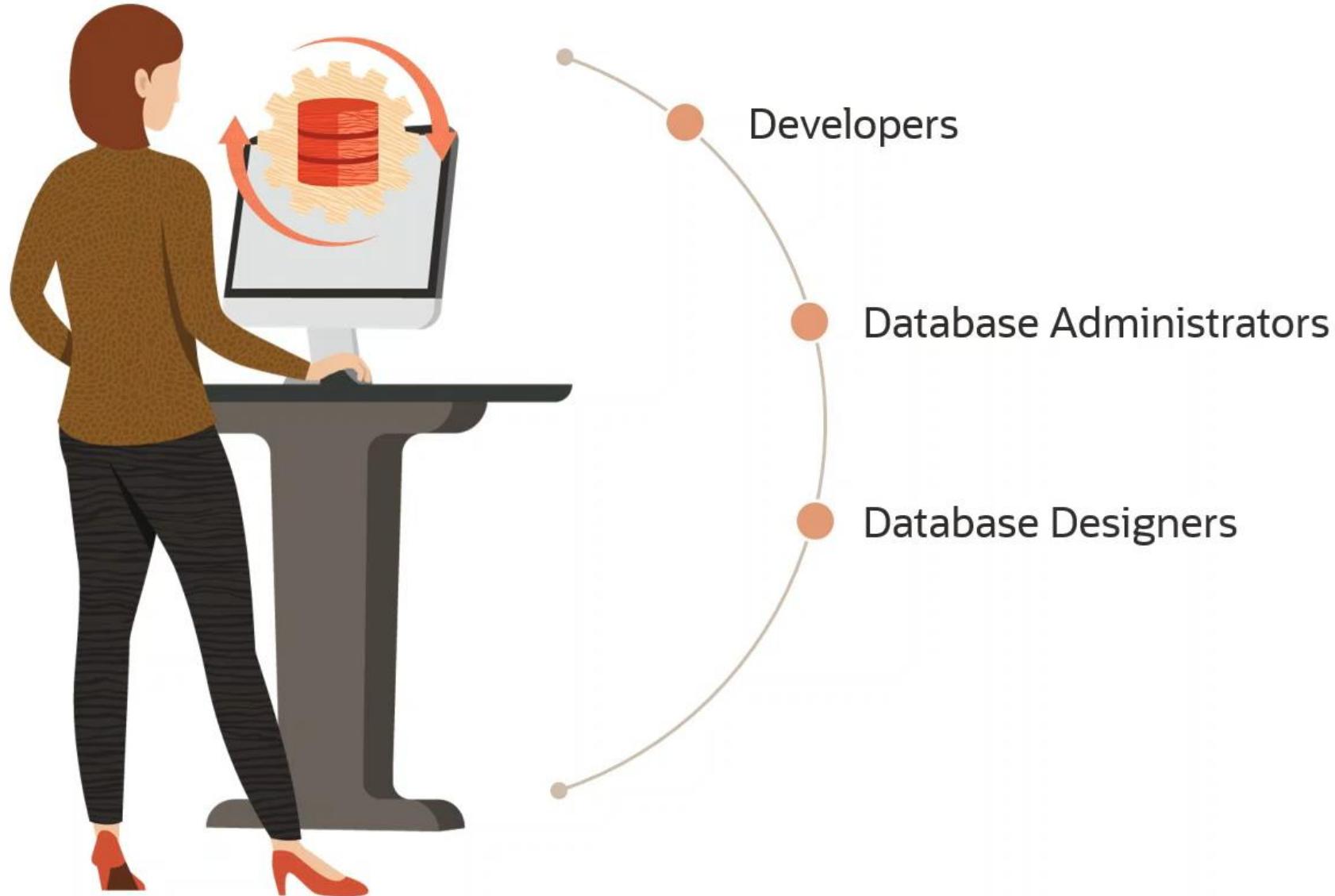
```
SELECT department_name  
FROM departments;
```

DEPARTMENT_NAME
Administration
Marketing
Shipping
IT
Sales
Executive
Accounting
Contracting

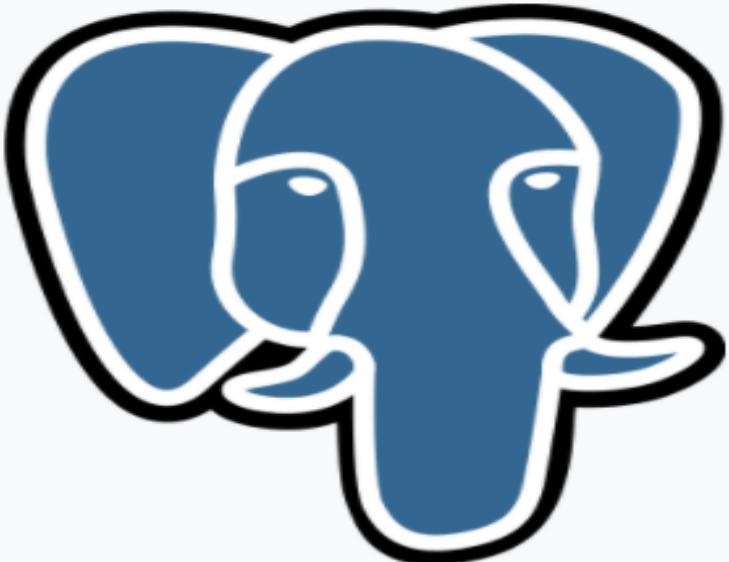
Oracle Server



Roles



PostgreSQL



*The World's Most Advanced Open Source
Relational Database^[1]*

Developer(s) PostgreSQL Global
Development Group^[2]

Initial release 8 July 1996;
25 years ago^[3]

Stable release 13.4^[4] / 12 August 2021;
31 days ago

Preview release 14 Beta 3 / 12 August
2021;
31 days ago^[5]

Repository [git.postgresql.org/gitweb/?
p=postgresql.git](https://git.postgresql.org/gitweb/?p=postgresql.git)

Written in C

Operating system macOS, Windows,^[6] Linux,



DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00
B01	PLANNING	000020	A00
C01	INFORMATION CENTER	000030	A00
D01	DEVELOPMENT CENTER		A00
E01	SUPPORT SERVICES	000050	A00
D11	MANUFACTURING SYSTEMS	000060	D01
D21	ADMINISTRATION SYSTEMS	000070	D01
E11	OPERATIONS	000090	E01
E21	SOFTWARE SUPPORT	000100	E01

- In this table we use:
- Columns—The ordered set of columns are DEPTNO, DEPTNAME, MGRNO, and ADMRDEPT. All the data in a given column must be of the same data type.
- Rows—Each row contains data for a single department and Row is also known as tuple.
- Values—at the intersection of a column and row is a value. For example, PLANNING is the value of the DEPTNAME column in the row for department B01.

**** RDBMS Vendors ****	**** RDBMS Product ****
Computer Associates	INGRES
IBM	DB2
INFORMIX Software	INFORMIX
Oracle Corporation	Oracle
Microsoft Corporation	MS Access
Microsoft Corporation	SQL Server
MySQL AB	MySQL
NCR Teradata	
PostgreSQL Dvlp Grp	PostgreSQL
Sybase	Sybase 11

SQL Syllabus

- Introduction
- What Database?
- What is table?
- What is tuple?
- Flavors & Vendors of databases?
- Who is DBA?
- What is Database Design and Database Architecture?

=====

SQL syllabus continued

=====

- SQL Commands Classification
- SQL SELECT
- SQL DISTINCT
- SQL WHERE
- SQL AND OR
- SQL IN
- SQL BETWEEN
- SQL Wildcard
- SQL LIKE
- SQL ILIKE
- SQL ORDER BY
- SQL GROUP BY
- SQL HAVING
- SQL ALIAS
- SQL AS
- SQL SELECT UNIQUE
- Playing with SQL

Database administrators (DBAs) design, write and take care of computer database systems so that the right person can get the information they need at the right time. Responsibilities of the job vary according to employment sector, but typically include:

- working with database software to find ways to store, organise and manage data
- troubleshooting
- keeping databases up to date
- helping with database design and development
- managing database access
- designing maintenance procedures and putting them into operation
- ensuring that databases meet user requirements
- liaising with programmers, applications/operational staff, IT project managers and other technical staff
- managing database security/integrity and backup procedures
- implementing security measures
- defining objectives through consultation with staff at all levels
- writing reports, documentation and operating manuals
- testing and modifying databases to ensure that they operate reliably
- providing user training, support and feedback
- writing disaster recovery plans
- archiving data.

Any organisation that stores large amounts of information and data may employ a database administrator.

Database Administrator

Database administrators manage how a software program is used by other employees. If a company or its employees need changes within a program's function, it is the responsibility of the database administrator adjust this. A database administrator is required to understand the needs of a company so they can organize a software program and its setup. Also, they oversee any issues a program may have and are responsible for addressing these issues to keep a system running efficiently. These professionals use codes to perform maintenance on databases and to build security. Information security is a top priority for companies and database administrators provide the IT team with a focus in managing security.

Job responsibilities of a database administrator include:

- Running tests to ensure a system runs properly
- Managing databases and permissions
- Repairing and backing up data within the systems to ensure data is maintained
- Performing mergers when a company transitions to a new system

Database Developer

Database developers maintain systems to ensure they run efficiently. They meet with clients to get information on how they need the system to function. Then, database developers build program designs based on those needs. They also manage these systems by removing old codes, writing new scripts to clean the system, and cleaning bugs that make their way into the system. These maintenance practices allow the system to stay clean, run at maximum speed and work to its full potential. Additionally, they monitor performance issues and troubleshoot any issues in the system.

Job responsibilities of a database developer include:

- Working in teams to develop the best solutions
- Analyzing current coding standards to design new strategies for a system
- Documenting database changes
- Writing reports on coding changes and proposing changes for the system
- Monitoring systems for issues

Database – Schema-table

Server which will have a Structured data

Database consists of **table** which helps us to manage data in structured format

Table is a entity which will have data in form of **ROWS** and **COLUMNS**

The screenshot shows a PostgreSQL terminal window. At the top, there is a command line input area with the following text:

```
1  SELECT * FROM public.products
2
```

Below the command line is a toolbar with several buttons: Explain, Messages, Notifications, Scratch Pad, and Data Output. A blue arrow points from the word "Data Output" to the "Data Output" button. The main area displays a table with the following data:

id integer	name character varying	price integer	description text
1	chocolate	10	[null]
2	chips	20	[null]
102	iphone11	60000	[null]
103	iphone10	50000	[null]
107	iphone7s	55000	[null]
108	iphone7	35000	[null]
109	iphone6	25000	[null]

No two columns can have same name

Every column will have datatypes

Primary keys

INTRODUCTION TO RELATIONAL DATABASES IN SQL



Every table should have PRIMARY KEY

PRIMARY KEY CANNOT BE NULL

PRIMARY KEY CANNOT DUPLICATE

PRIMARY KEY WILL BE ALWAYS DISTINCT

Primary keys

- One primary key per database table, chosen from candidate keys
- Uniquely identifies records, e.g. for referencing in other tables
- Unique and not-null constraints both apply
- Primary keys are time-invariant: choose columns wisely!

Specifying primary keys

```
CREATE TABLE products (
    product_no integer UNIQUE NOT NULL,
    name text,
    price numeric
);
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
```

```
CREATE TABLE example (
    a integer,
    b integer,
    c integer,
    PRIMARY KEY (a, c)
);
```

Taken from the [PostgreSQL documentation](#).

Specifying primary keys (contd.)

```
ALTER TABLE table_name  
ADD CONSTRAINT some_name PRIMARY KEY (column_name)
```

Identify the primary key

Have a look at the example table from the previous video. As the database designer, you have to make a wise choice as to which column should be the primary key.

license_no	serial_no	make	model	year
Texas ABC-739	A69352	Ford	Mustang	2
Florida TVP-347	B43696	Oldsmobile	Cutlass	5
New York MPO-22	X83554	Oldsmobile	Delta	1
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	4
Texas RSK-629	U028365	Jaguar	XJS	4

Which of the following column or column combinations could best serve as primary key?

Surrogate keys

INTRODUCTION TO RELATIONAL DATABASES IN SQL

SQL

Surrogate keys

- Primary keys should be built from as few columns as possible
- Primary keys should never change over time

license_no	serial_no	make	model	color
Texas ABC-739	A69352	Ford	Mustang	blue
Florida TVP-347	B43696	Oldsmobile	Cutlass	black
New York MPO-22	X83554	Oldsmobile	Delta	silver
California 432-TFY	C43742	Mercedes	190-D	champagne
California RSK-629	Y82935	Toyota	Camry	red
Texas RSK-629	U028365	Jaguar	XJS	blue

make	model	color
Ford	Mustang	blue
Oldsmobile	Cutlass	black
Oldsmobile	Delta	silver
Mercedes	190-D	champagne
Toyota	Camry	red
Jaguar	XJS	blue

Adding a surrogate key with serial data type

```
ALTER TABLE cars
ADD COLUMN id serial PRIMARY KEY;
INSERT INTO cars
VALUES ('Volkswagen', 'Blitz', 'black');
```

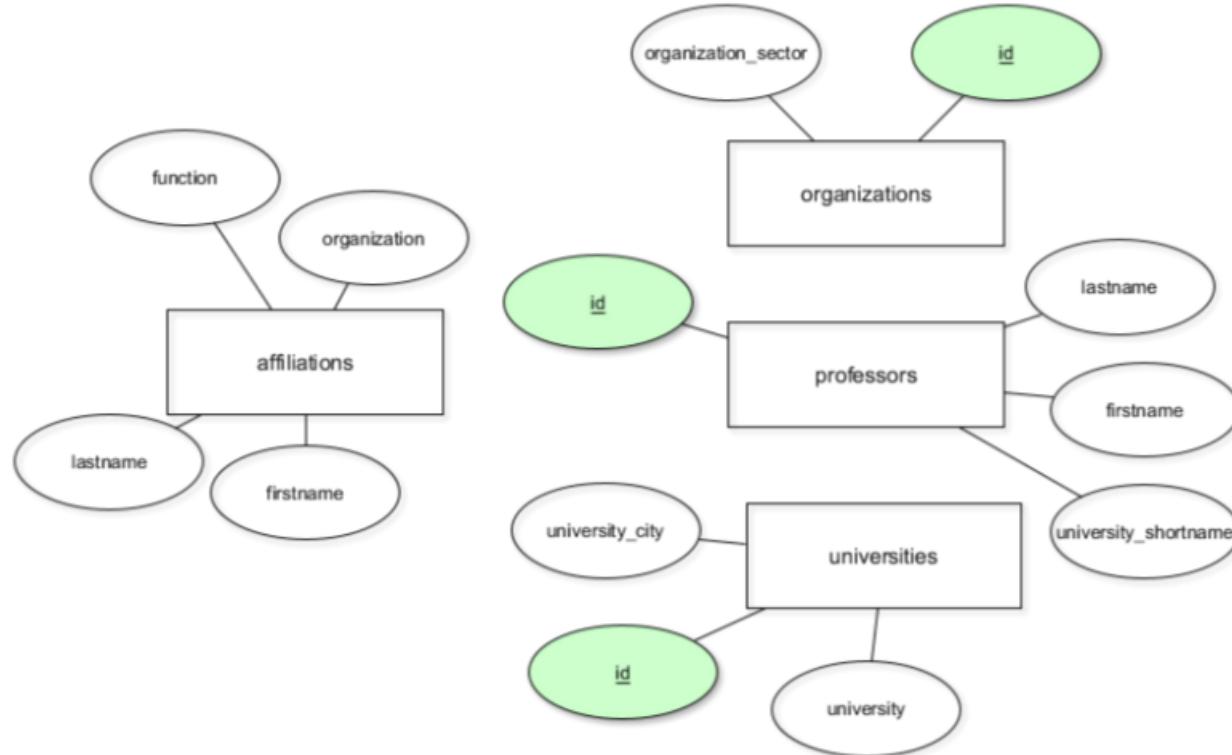
make	model	color	id
Ford	Mustang	blue	1
Oldsmobile	Cutlass	black	2
Oldsmobile	Delta	silver	3
Mercedes	190-D	champagne	4
Toyota	Camry	red	5
Jaguar	XJS	blue	6
Volkswagen	Blitz	black	7

Adding a surrogate key with serial data type (contd.)

```
INSERT INTO cars  
VALUES ('Opel', 'Astra', 'green', 1);
```

```
duplicate key value violates unique constraint "id_pkey"  
DETAIL: Key (id)=(1) already exists.
```

- "id" uniquely identifies records in the table – useful for referencing!



Test your knowledge before advancing

Before you move on to the next chapter, let's quickly review what you've learned so far about attributes and key constraints. If you're unsure about the answer, please quickly review chapters 2 and 3, respectively.

Let's think of an entity type "student". A student has:

- a **last name** consisting of *up to 128* characters (required),
 - a unique **social security number**, consisting only of integers, that should serve as a key,
 - a **phone number** of *fixed length 12*, consisting of numbers and characters (but some students don't have one).
-
- Given the above description of a student entity, create a table `students` with the correct column types.
 - Add a `PRIMARY KEY` for the social security number `ssn`.

*Note that there is no formal length requirement for the integer column.
The application would have to make sure it's a correct SSN!*

query.sql

solution.sql

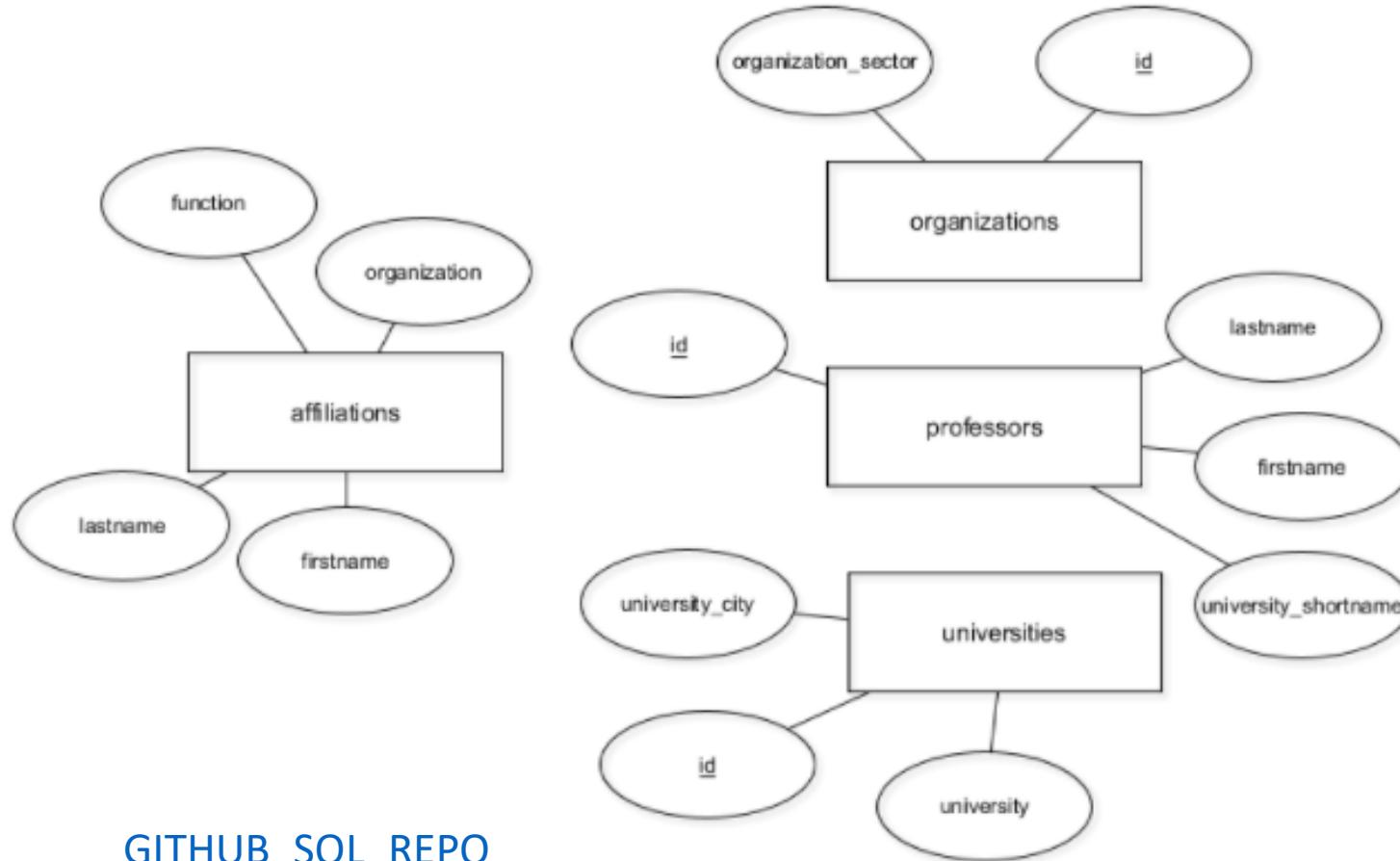
```
1 -- Create the table
2 CREATE TABLE students (
3     last_name varchar(128) NOT NULL,
4     ssn integer PRIMARY KEY,
5     phone_no char(12)
6 );
```

Model 1:N relationships with foreign keys

INTRODUCTION TO RELATIONAL DATABASES IN SQL

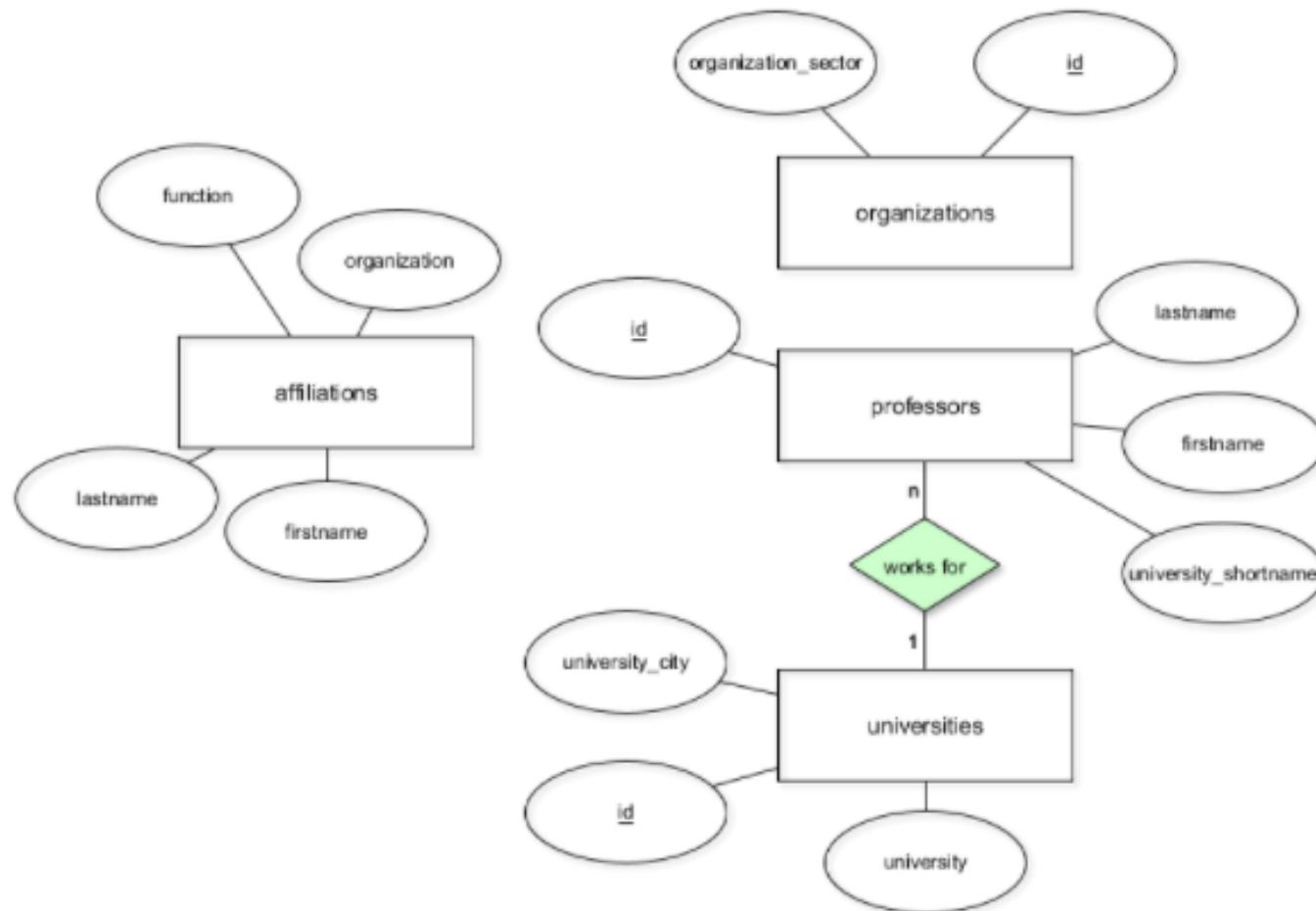


The current database model



[GITHUB SQL REPO](#)

The next database model



Implementing relationships with foreign keys

- A foreign key (FK) points to the primary key (PK) of another table
- Domain of FK must be equal to domain of PK
- Each value of FK must exist in PK of the other table (FK constraint or "referential integrity")
- FKS are not actual *keys*

[GITHUB SQL REPO](#)

A query

```
SELECT * FROM professors LIMIT 8;
```

<code>id</code>	<code>firstname</code>	<code>lastname</code>	<code>university_s..</code>
1	Karl	Aberer	EPF
2	Reza Shokrollah	Abhari	ETH
3	Georges	Abou Jaoudé	EPF
4	Hugues	Abriel	UBE
5	Daniel	Aebersold	UBE
6	Marcelo	Aebi	ULA
7	Christoph	Aebi	UBE
8	Patrick	Aebischer	EPF

```
SELECT * FROM universities;
```

<code>id</code>	<code>university</code>	<code>university_city</code>
EPF	ETH Lausanne	Lausanne
ETH	ETH Zürich	Zurich
UBA	Uni Basel	Basel
UBE	Uni Bern	Bern
UFR	Uni Freiburg	Fribourg
UGE	Uni Genf	Geneva
ULA	Uni Lausanne	Lausanne
UNE	Uni Neuenburg	Neuchâtel
USG	Uni St. Gallen	Saint Gallen
USI	USI Lugano	Lugano
UZH	Uni Zürich	Zurich

[GITHUB SQL REPO](#)

Specifying foreign keys

```
CREATE TABLE manufacturers (
    name varchar(255) PRIMARY KEY);

INSERT INTO manufacturers
VALUES ('Ford'), ('VW'), ('GM');

CREATE TABLE cars (
    model varchar(255) PRIMARY KEY,
    manufacturer_name varchar(255) REFERENCES manufacturers (name));

INSERT INTO cars
VALUES ('Ranger', 'Ford'), ('Beetle', 'VW');
```

[GITHUB SQL REPO](#)

Specifying foreign keys

```
CREATE TABLE manufacturers (
    name varchar(255) PRIMARY KEY);

INSERT INTO manufacturers
VALUES ('Ford'), ('VW'), ('GM');

CREATE TABLE cars (
    model varchar(255) PRIMARY KEY,
    manufacturer_name varchar(255) REFERENCES manufacturers (name);

INSERT INTO cars
VALUES ('Ranger', 'Ford'), ('Beetle', 'VW');
```

```
-- Throws an error!
INSERT INTO cars
VALUES ('Tundra', 'Toyota');
```

[GITHUB_SQL_REPO](#)

Specifying foreign keys to existing tables

```
ALTER TABLE a  
ADD CONSTRAINT a_fkey FOREIGN KEY (b_id) REFERENCES b (id);
```

[GITHUB SQL REPO](#)

```
-- players(id,name,is_captian,is_vice_captian)
-- teams (id,name,coach)
-- team_players(id,team_id,player_id)
--matches(id,toss_owned_by(team_id),date,team1_id,team2_id,venue_id,start_timing,end_timing,third_umpire)
-- venues
-- sponsors
-- umpires
-- commentator
```

[GITHUB_SQL_REPO](#)

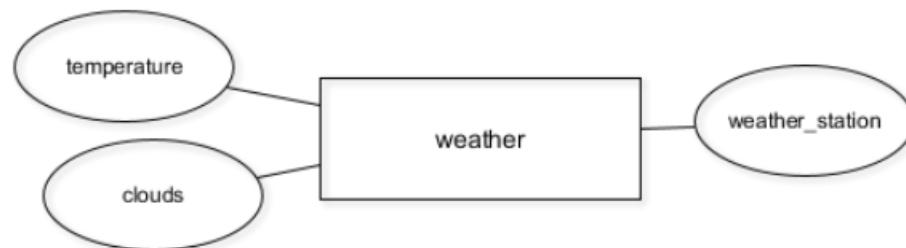
SYNTAX OF CREATE TABLE

Query Editor Query History

```
1 CREATE TABLE table_name(col1 data type,col2 data type,  
2 .  
3 .  
4 .  
5 .  
6 .  
7 .coln data type);
```

Create new tables with CREATE TABLE

```
CREATE TABLE weather (
    clouds text,
    temperature numeric,
    weather_station char(5)
);
```



```
1 CREATE TABLE ipl2021.teams(id serial primary key,  
2                             team_name character varying(100) not null,  
3                             owned_by integer,  
4                             established_in date,  
5                             is_active boolean,  
6                             total_cost numeric,  
7                             team_size integer,  
8                             head_coach integer);  
9
```

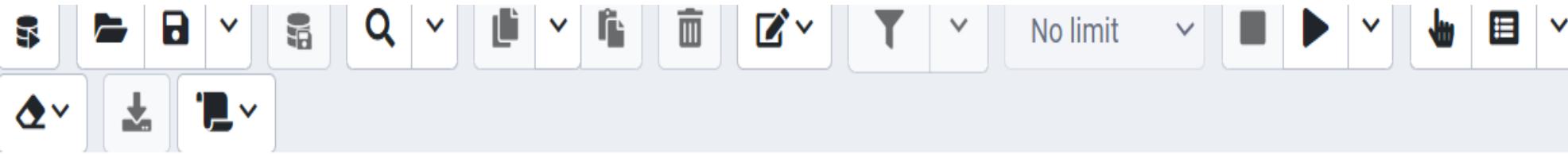
Explain Messages Notifications Scratch Pad Data Output

CREATE TABLE

Query returned successfully in 57 msec.

✓ Query returned successfully in 57 msec.

Insert syntax in sql



The screenshot shows the MySQL Workbench interface. The top bar contains various icons for database management, such as connection, folder, and search. Below the toolbar is a menu bar with options like 'File', 'Edit', 'View', 'Tools', 'Help', and a 'Query' dropdown. The main window has tabs for 'Query Editor' (which is selected) and 'Query History'. The code area displays the following SQL query:

```
10 INSERT INTO table_name(col1,col2,col3.....,coln) values(val1,val2,val3.....,vln);  
11  
12  
13
```

SINGLE ROW INSERTION

Query Editor Query History

```
10  INSERT INTO ipl2021.teams(team_name,established_in) values('RCB',null);  
11  
12  
13  select * from ipl2021.teams  
14  
15  
16  
17  
18
```

Explain Messages Notifications Scratch Pad Data Output

	id [PK] integer	team_name character varying (100)	owned_by integer	established_in date	is_active boolean	total_cost numeric	team_size integer	head_coach integer	
1	1	RCB	[null]	[null]	[null]	[null]	[null]	[null]	

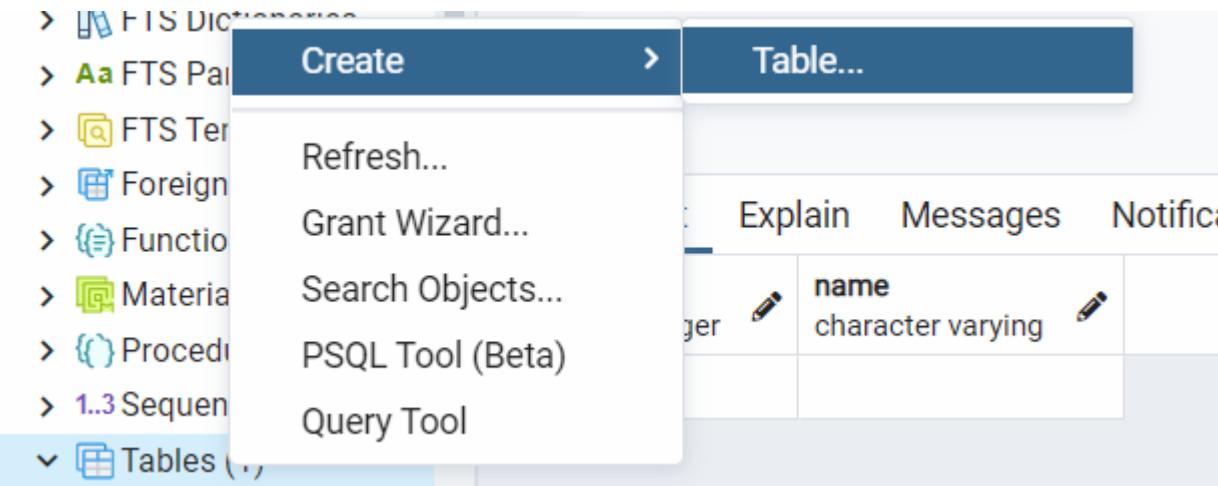
Query Editor Query History

```
10 INSERT INTO ipl2021.teams(team_name,established_in,is_active,total_cost,team_size)
11 values
12 ('CSK','01-01-2008',true,1000,20),
13 ('SRH','01-01-2008',true,1000,20),
14 ('MI','01-01-2008',true,1000,20),
15 ('DC','01-01-2008',true,1000,20),
16 ('KKR','01-01-2008',true,1000,20);
17
18
```

Explain Messages Notifications Scratch Pad Data Output

INSERT 0 5

Query returned successfully in 121 msec.



Query Editor Query History

1 CREATE TABLE DEPARTMENT(**ID** SERIAL,**DNAME** CHARACTER VARYING);

Data Output Explain **Messages** Notifications Scratch Pad

CREATE TABLE

Query returned successfully in 88 msec.

```
CREATE TABLE ORAGANIZATIONS(
    ID SERIAL,
    ORG_NAME CHARACTER VARYING,
    ADDRESS CHARACTER VARYING,
    GSTIN CHARACTER VARYING(30),
    PHONENUM INTEGER,
    EMAIL CHARACTER VARYING,
    CEO CHARACTER VARYING,
    COMPANY_SIZE INTEGER,
    COMP_VALUE CHARACTER VARYING,
    EMPLOYEE_STRENGTH INTEGER,
    TECHNOLOGY CHARACTER VARYING,
    CATEGORY CHARACTER VARYING,
    WEBSITE_URL CHARACTER VARYING,
    HEAD_OFFICE CHARACTER VARYING,
    LANDLINE CHARACTER VARYING

)
```

Data types in postgresql

Character varying

Integer

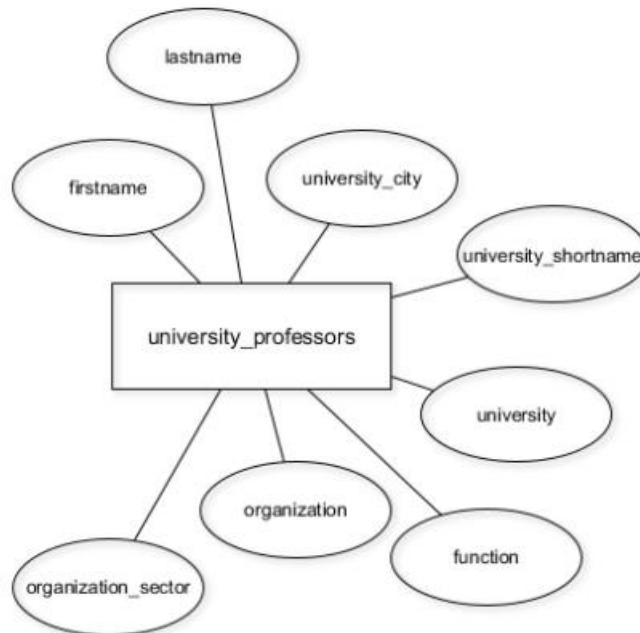
Numeric -- 3.14

Date

[Youtube Videos](#)

SQL

Currently: One "entity type" in the database



A better database model with three entity types

Old:



New:



Update your database as the structure changes

INTRODUCTION TO RELATIONAL DATABASES IN SQL

Only store DISTINCT data in the new tables

```
SELECT COUNT(*)  
FROM university_professors;
```

```
count  
-----  
1377
```

```
SELECT COUNT(DISTINCT organization)  
FROM university_professors;
```

```
count  
-----  
1287
```

RENAME a COLUMN in affiliations

```
CREATE TABLE affiliations (
    firstname text,
    lastname text,
    university_shortname text,
    function text,
    organisation text
);
```

```
ALTER TABLE table_name
RENAME COLUMN old_name TO new_name;
```

DROP a COLUMN in affiliations

```
CREATE TABLE affiliations (
    firstname text,
    lastname text,
    university_shortname text,
    function text,
    organization text
);
```

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Our motivating question:

How should we organize and manage data?

- **Schemas:** *How should my data be logically organized?*
- **Normalization:** *Should my data have minimal dependency and redundancy?*
- **Views:** *What joins will be done most often?*
- **Access control:** *Should all users of the data have the same level of access*
- **DBMS:** *How do I pick between all the SQL and noSQL options?*
- and more!

Structuring data

1. Structured data

- Follows a schema
- Defined data types & relationships

_e.g., SQL, tables in a relational database _

2. Unstructured data

- Schemaless
- Makes up most of data in the world

e.g., photos, chat logs, MP3

3. Semi-structured data

- Does not follow larger schema
- Self-describing structure

e.g., NoSQL, XML, JSON

```
1 INSERT INTO capgemini.oraganizations(
2     org_name, address, gstin, phonenum, email, ceo,
3     company_size, comp_value, employee_strength, technology,
4     category, website_url, head_office, landline)
5     VALUES ('x','a');
```

Data Output Explain Messages Notifications Scratch Pad

ERROR: INSERT has more target columns than expressions

LINE 2: org_name, address, gstin, phonenum, email, ceo,
 ^

SQL state: 42601

Character: 60

[Query Editor](#) [Query History](#)

```
1 INSERT INTO capgemini.oraganizations(  
2     org_name, address)  
3     VALUES ('CTS','Bangalrore');
```

[Query Editor](#) [Query History](#)

```
1 SELECT * FROM capgemini.oraganizations  
2
```

[Data Output](#) [Explain](#) [Messages](#) [Notifications](#) [Scratch](#)

INSERT 0 1

Query returned successfully in 68 msec.

[Data Output](#) [Explain](#) [Messages](#) [Notifications](#) [Scratch Pad](#)

	id integer	org_name character varying	address character varying	gstin character varying (30)	phonenum integer
1	1	CTS	Bangalrore	[null]	[null]

Inserting data into different ways

```
5  
6 insert into department(department_name) values('Admin');  
7  
8 insert into department(department_name) Values('Technical'),('Finance');  
9
```

Data Output Explain Messages Notifications Scratch Pad

INSERT 0 2

Query returned successfully in 73 msec

Never use “SQL KEYWORD” as column name

How to view entire table data

Query Editor Query History

1 `select * from products`

Data Output Explain Messages Notifications Scratch Pad

	id integer	name character varying	price integer	description text	title character varying (100)	is_deleted boolean
1	1	chocolate	10	[null]	[null]	false
2	2	chips	20	[null]	[null]	false
3	102	iphone11	60000	[null]	[null]	false
4	103	iphone10	50000	[null]	[null]	false
5	104	iphone9	40000	[null]	[null]	false
6	105	iphone8	38000	[null]	[null]	false
7	106	iphone9s	55000	[null]	[null]	false
8	107	iphone7s	55000	[null]	[null]	false
9	108	iphone7	35000	[null]	[null]	false
10	109	LG-Laptop	35000	[null]	[null]	false
11	110	Samsung-Laptop	35000	[null]	[null]	false
12	111	MacBook	90000	[null]	[null]	false
13	112	MacBookAir	80000	[null]	[null]	false
14	113	HP-ProBook	85000	[null]	[null]	false

✓ Successfully run. Total query runtime: 97 msec. 32 rows affected.

Learning to COUNT

What if you want to count the number of employees in your employees table? The `COUNT()` function lets you do this by returning the number of rows in one or more columns.

For example, this code gives the number of rows in the `people` table:

```
SELECT COUNT(*)  
FROM people;
```

Query Editor Query History

```
1 select count(1) from products  
2  
3  
4  
5 count of rows
```

Data Output Explain Messages Notifications Scratch Pad

	count	bigint
1	32	

Query Editor Query History

```
1  SELECT DISTINCT price from products  
2  
3  
4  
5
```

Data Output Explain Messages Notifications Scratch Pad

	price integer 
1	1875000
2	100000
3	50000
4	85000
5	375000
6	75000
7	1075000
8	10000
9	40000
10	80000
11	30
12	64000
13	60000
14	90000

COUNT WITH DISTINCT

Query Editor Query History

```
1 SELECT count(DISTINCT price) from products
2
3
4
5
```

Data Output Explain Messages Notifications Scratch Pad

	count	bigint
1	26	

What is the difference between
count () and count(distinct any column)

Only store DISTINCT data in the new tables

```
SELECT COUNT(*)  
FROM university_professors;
```

```
count  
-----  
1377
```

```
SELECT COUNT(DISTINCT organization)  
FROM university_professors;
```

```
count  
-----  
1287
```

Update your database as the structure changes

INTRODUCTION TO RELATIONAL DATABASES IN SQL



```
ALTER TABLE table_name  
ADD COLUMN column_name  
DATA_TYPE;
```

```
ALTER TABLE table_name  
RENAME COLUMN  
old_column_name to  
new_column_name;
```

Query Editor Query History

```
1 select * from department;  
2  
3  
4 ALTER TABLE department RENAME COLUMN dname to department_name;  
5
```

```
ALTER TABLE table_name  
DROP column_name ;
```

```
ALTER TABLE table_name  
DROP Column column_name ;
```

```
ALTER TABLE table_name ADD  
CONSTRAINT constraint_name  
PRIMARY KEY (column(s));
```

What is primary key ?

What is composite key ?

Composite key Constraint

```
41  
42 create table courses(id serial, course_name character varying UNIQUE ,price integer default 10000  
43 ,CHECK (price>=5000),primary key(course_name,price));  
44  
45 insert into courses(course_name) values('java'),('sql');  
46  
47 insert into courses(course_name,price) values('java',1000),('c',500);  
48  
49
```

Data Output Explain Messages Notifications Scratch Pad

ERROR: duplicate key value violates unique constraint "courses_pkey"
DETAIL: Key (course_name, price)=(java, 10000) already exists.
SQL state: 23505

```
create table staff(id serial,  
staff_name character varying  
,department_id integer  
REFERENCES department(id));
```

Query Editor Query History

```
1 select * from department;  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13
```

Data Output Explain Messages Notifi

	id [PK] integer	department_name character varying	
1	1	Admin	
2	2	Technical	
3	3	Finance	

37

```
38 insert into staff(staff_name,department_id) values('sunitha',2),('chaitanya',3),('Jyoshna',10);
```

39

40

41

Data Output Explain **Messages** Notifications Scratch Pad

ERROR: insert or update on table "staff" violates foreign key constraint "staff_department_id_fkey"

DETAIL: Key (department_id)=(10) is not present in table "department".

SQL state: 23503

Can you insert duplicate data when we have primary key on column

```
11  insert into department(id,department_name) values(1,'Admin');  
12  
13  
14  
15  
16  
17 |
```

Data Output Explain Messages Notifications Scratch Pad

ERROR: duplicate key value violates unique constraint "pk_dept"
DETAIL: Key (id)=(1) already exists.
SQL state: 23505

RENAME and DROP COLUMNs

You'll use the following queries:

- To rename columns:

```
ALTER TABLE table_name  
    RENAME COLUMN old_name TO new_name;
```

- To delete columns:

```
ALTER TABLE table_name  
    DROP COLUMN column_name;
```

```
23 TRUNCATE TABLE DEPARTMENT  
24 |  
25  
26  
27  
28
```

Data Output Explain Messages Notifications Scratch Pad

ERROR: cannot truncate a table referenced in a foreign key constraint

DETAIL: Table "staff" references "department".

HINT: Truncate table "staff" at the same time, or use TRUNCATE ... CASCADE.

SQL state: 0A000

Can a foreign key contain null values?

Can a foreign references a non-primary key ?

Select a limited number of records from the `films` table.

Complete the code to return the output

```
SELECT title
```

?	?
?	?

title
Intolerance: Love's Struggle Throughout the Ages
Over the Hill to the Poorhouse
The Big Parade
Metropolis
Pandora's Box

Fill in the blanks

films 1

LIMIT 2

SELECT 3

3 4

FROM 5

5 6

Which one of these is a valid statement?

Select the correct answer

SELECT COUNT(*) FROM DISTINCT states

PRESS 1

SELECT DISTINCT(COUNT year) FROM states

PRESS 2

SELECT COUNT(*) FROM states

PRESS 3

Complete the code to return the output

```
SELECT ?  
FROM states;
```

id	state	year	total_revenue
1	ALABAMA	1992	2678885
2	ALASKA	1992	1049591
3	ARIZONA	1993	3427976
4	CALIFORNIA	1994	23440845
5	COLORADO	1994	3061865
6	FLORIDA	1995	13242710
7	INDIANA	1992	5060274
8	KANSAS	1996	2914544
9	MICHIGAN	1996	13012080
10	MONTANA	1996	944623

Fill in the blanks

all 1 ** 2 * 3 columns 4

Complete the code to return the output

```
SELECT name, country  
FROM companies  
WHERE name [ ] ('Shopify', 'Root', 'Software', 'Delta', 'Oasis');
```

name	country
Shopify	USA
Root	USA
Software	USA
Delta	USA
Oasis	USA

Fill in the blanks

IN 1 BETWEEN 2 COUNT 3

How would you find the records for all Spanish films released before 2000?

Complete the code to return the output

```
SELECT title, language  
FROM films  
WHERE  
    language = ?  
    AND release_year ? 2000  
LIMIT 5
```

title	language
El Mariachi	Spanish
La otra conquista	Spanish
Tango	Spanish

Fill in the blanks

'Spanish' 1 = 2 Spanish 3 < 4 AND 5 release_year 6

What is the result of `SELECT (9 / 2); ?`

Select the correct answer

9 / 2

PRESS **1**

4

PRESS **2**

4.5

PRESS **3**

4.0

PRESS **4**

Complete the code to return the output

```
SELECT name  
FROM companies  
ORDER BY name ?  
LIMIT 5;
```

name
Wire
Tesla
Statix
Software
Shopify

Fill in the blanks

DESCEND 1

REVERSE 2

DESC 3

SQL Aggregate Functions

- 1)Count
- 2)Avg
- 3)Max
- 4)Min
- 5)SUM

Data	python	postgresql
Integer data	int	Integer/int /bigint
Floating point data	float	numeric
String data	String	Character varying /char
	bool	boolean
	date	DATE
Auto genaration	Int(mapped to primary key in table)	SERIAL

Keywords in SQL

CREATE	ALTER	COLUMN	IN	DISTINCT
INSERT	ADD	NOT NULL	PRIMARY KEY	LIKE
SELECT	WHERE	UNIQUE	SERIAL	ILIKE
UPDATE	FROM	DEFAULT	JOIN	
TRUNCATE	INTO	CHECK	INNER JOIN	
DROP	VALUES	CONSTRAINT	RIGHT JOIN	
DELETE	SET	AND	LEFT JOIN	
ORDER	TABLE	OR	AS	

Inbuilt functions in SQL

Lower				
upper				
rank				
max				
min				
avg				
sum				
count				

Retail_store

Data Output					
	id	store_name	sales	txn_date	
1	1	Bangalore	1500	2021-01-01	
2	2	Hyderabad	500	2021-01-01	
3	4	Delhi	600	2021-01-07	
4	3	Delhi	400	2021-01-05	

Retail_store_transactions

Data Output					
	id	store_name	sales	txn_date	
1	1	Bangalore	1500	2021-01-01	
2	2	Hyderabad	500	2021-01-01	
3	4	Delhi	600	2021-01-07	
4	3	Delhi	400	2021-01-05	
5	5	Bangalore	500	2021-01-19	
6	6	Hyderabad	1000	2021-01-19	

```
5 -- TO FECTCH RECORDS IN EVERY CITY ON WHICH DATE MAX SALES ARE THERE
6 select store_name,max(sales) from retail_store
7
8
9
10
11
12
13
14
15
16
```

Data Output Explain Messages Notifications

ERROR: column "retail_store.store_name" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: select store_name,max(sales) from retail_store
 ^
SQL state: 42803
Character: 8

Query Editor Query History

```
4 |
5 -- TO FECTCH RECORDS IN EVERY CITY IN WHIHC MAX SALES ARE THERE
6 select store_name,max(sales) from retail_store GROUP BY store_name
7
8 -- TO FECTCH RECORDS IN EVERY CITY ON WHICH DATE MAX SALES ARE THERE
9 select txn_date,max(sales) from retail_store GROUP BY txn_date
10
11 -- TO FECTCH RECORDS IN EVERY CITY ON WHICH DATE and CITY MAX SALES ARE THERE
12 select store_name,txn_date,max(sales) from retail_store GROUP BY store_name,txn_date
13
14 -- TO FECTCH RECORDS IN EVERY CITY ON WHICH DATE MIN SALES ARE THERE
15 select txn_date,min(sales) from retail_store GROUP BY txn_date
16
17 -- TO FECTCH RECORDS IN EVERY CITY IN WHICH MIN SALES ARE THERE
18 select store_name,min(sales) from retail_store GROUP BY store_name
19
20
```

Data Output Explain Messages Notifications

	store_name character varying (100)	min integer	
1	Delhi	400	
2	Hyderabad	500	
3	Bangalore	500	

[Query Editor](#) [Query History](#)

```
23 -- FIND AVG SALES CITY WISE IN RETAIL STORE
24 SELECT store_name,AVG(SALES) FROM retail_store GROUP BY store_name
25
26 -- FIND AVG OF SALES IN FIFTEEN DAYS
27 SELECT AVG(sales) FROM retail_store WHERE txn_date between '01-01-2021' and '30-01-2021'
28 |
29 SELECT AVG(sales) FROM retail_store WHERE txn_date >='01-01-2021' and txn_date<='15-01-2021'
30
31 -- FIND AVG SALES CITY WISE IN RETAIL STORE WITH DESCENDING ORDER
32 SELECT AVG(SALES) FROM retail_store GROUP BY SALES ORDER BY SALES DESC
33
34 --FIND DISTINCT CITIES IN RETAIL STORE
35 SELECT DISTINCT store_name FROM retail_store
36
37 -- FIND TOPPEST CITY WITH MAX SALES
38 SELECT store_name,max(sales) FROM retail_store GROUP BY store_name,SALES ORDER BY SALES DESC LIMIT 1
```

[Data Output](#) [Explain](#) [Messages](#) [Notifications](#)

	store_name	max	
	character varying (100)	integer	
1	Bangalore	1500	

Group by is used when we use aggregate functions with other columns

Order by default order is Ascending order

SELECT , COLs, AGGREGATE FUNC , FROM , WHERE , GROUP BY , ORDER BY , LIMIT 1

```
3 select store_name,avg(sales)::numeric as sales from retail_store order by sales GROUP BY store_name DESC LIMIT 1
```

Data Output Explain Messages Notifications Query History

ERROR: syntax error at or near "GROUP"
LINE 1: ...numeric as sales from retail_store order by sales GROUP BY s...
^

SQL state: 42601

Character: 81

Count function in sql

```
1 select count(*),store_name from retail_store group by store_name
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

```
10
```

```
11
```

```
12
```

```
13
```

Data Output

Explain

Messages

Notifications

Query History

	count	store_name
	bigint	character varying (100)
1	3	Pune
2	2	Delhi
3	2	Hyderabad
4	2	Bangalore

AND & OR Condition in SQL

```
test/postgres@PostgreSQL 13 ~
Query Editor Query History
1 select * from retail_store
2
3 --AND & OR conditions
4 -- SELECT COL_NAME FROM TABLE_NAME WHERE COND1 {AND|OR} COND2;
5
6 -- FETCH THE CITIES WHOSE SALES > 1000 or WHOSE SALES <500
7 SELECT store_name FROM retail_store WHERE sales >1000 OR sales<500
8
9 -- AND
10 SELECT * FROM employee WHERE first_name='Teja' and last_name='Bandaru'
11
12 -- FETCH SALES on JAN 1st OR JAN 7th
13 SELECT SALES,txn_date FROM retail_store WHERE txn_date ='01-01-2021' or txn_date='07-01-2021'

Data Output Explain Messages Notifications


|   | sales<br>integer | txn_date<br>date |
|---|------------------|------------------|
| 1 | 1500             | 2021-01-01       |
| 2 | 500              | 2021-01-01       |
| 3 | 600              | 2021-01-07       |


```

```
CREATE TABLE tablename(col1 data type,  
                     col2 data type , .... coln data type );
```

A table can have ‘n’ number of columns and ‘n’ number of rows

UNIQUE CONSTRAINT

test postgres@PostgreSQL 13

Query Editor Query History

```
1 CREATE TABLE besant.employees(id serial,
2                               first_name character varying(50) NOT NULL,
3                               second_name character varying(20) NOT NULL,
4                               dob date,
5                               mobile_no bigint UNIQUE NOT NULL,
6                               email_id character varying(50) UNIQUE );
7
8
9 INSERT INTO besant.employees(first_name,second_name,dob,mobile_no,email_id)
10 values('Scott','styris','11-02-1989',9899898898,'scott@gmail.com');
11
12 INSERT INTO besant.employees(first_name,second_name,dob,mobile_no,email_id)
13 values('Jacob','Oram','11-02-1989',9899898898,'jacob@gmail.com');
14
15
16
17
```

Data Output Explain Messages Notifications

ERROR: duplicate key value violates unique constraint "employees_mobile_no_key"
DETAIL: Key (mobile_no)=(9899898898) already exists.
SQL state: 23505

```
"ALTER TABLE "table_name" ADD "column_name" DATA TYPE"
```

```
ALTER TABLE besant.employees RENAME COLUMN sex to GENDER ;
```

```
CREATE TABLE employee_constraint(ID serial PRIMARY KEY ,  
                                first_name character varying(100) NOT NULL,  
                                last_name character varying(50) NOT NULL,  
                                mobile bigint UNIQUE,  
                                is_laptop_tagged boolean DEFAULT false  
                                );
```

```
INSERT INTO employee_constraint(first_name,last_name,mobile,is_laptop_tagged) values  
('Fatima','begum',9878987888,TRUE);
```

```
INSERT INTO employee_constraint(first_name,last_name,mobile,is_laptop_tagged) values ('Fatima','khan',9878987988,TRUE);
```

```
INSERT INTO employee_constraint(first_name,last_name,mobile) values ('Rafiq','mohammed',9878987788);
```

Introduction to NULL and IS NULL

In SQL, `NULL` represents a missing or unknown value. You can check for `NULL` values using the expression `IS NULL`. For example, to count the number of missing birth dates in the `people` table:

```
SELECT COUNT(*)
FROM people
WHERE birthdate IS NULL;
```

As you can see, `IS NULL` is useful when combined with `WHERE` to figure out what data you're missing.

Sometimes, you'll want to filter out missing values so you only get results which are not `NULL`. To do this, you can use the `IS NOT NULL` operator.

For example, this query gives the names of all people whose birth dates are *not* missing in the `people` table.

```
SELECT name
FROM people
WHERE birthdate IS NOT NULL;
```

Import and export of data Into Database

Query Editor

```
1 create table cities(name character varying, country_code character varying,  
2                      city_proper_pop character varying, metro_pop character varying,  
3                      urbanarea_pop character varying)
```

Data Output Explain Messages Notifications Query History

CREATE TABLE

Query returned successfully in 72 msec.

 cities - Notepad

File Edit Format View Help

name,country_code,city_proper_pop,metroarea_pop,urbanarea_pop

Abidjan,CIV,4765000,0,4765000

Abu Dhabi,ARE,1145000,0,1145000

Abuja,NGA,1235880,6000000,1235880

Accra,GHA,2070460,4010050,2070460

Addis Ababa,ETH,3103670,4567860,3103670

Ahmedabad,IND,5570580,0,5570580

Alexandria,EGY,4616620,0,4616620

Algiers,DZA,3415810,5000000,3415810

Almaty,KAZ,1703480,0,1703480

Ankara,TUR,5271000,4585000,5271000

Auckland,NZL,1495000,1614300,1495000

Baghdad,IRQ,7180890,0,7180890

Baku,AZE,3202300,4308740,3202300

Bandung,IDN,2575480,6965660,2575480

Bangkok,THA,8280920,14998000,8280920

Barcelona,ESP,1604560,5375770,1604560

Baranquilla,COL,1386860,2370750,1386860

Basra,IRQ,2750000,0,2750000

Beijing,CHN,21516000,24900000,21516000

Belo Horizonte,BRA,2502560,5156220,2502560

Bengaluru,IND,8425970,9807000,8425970

Berlin,DEU,3517420,5871020,3517420

Bhopal,IND,1798220,1864390,1798220

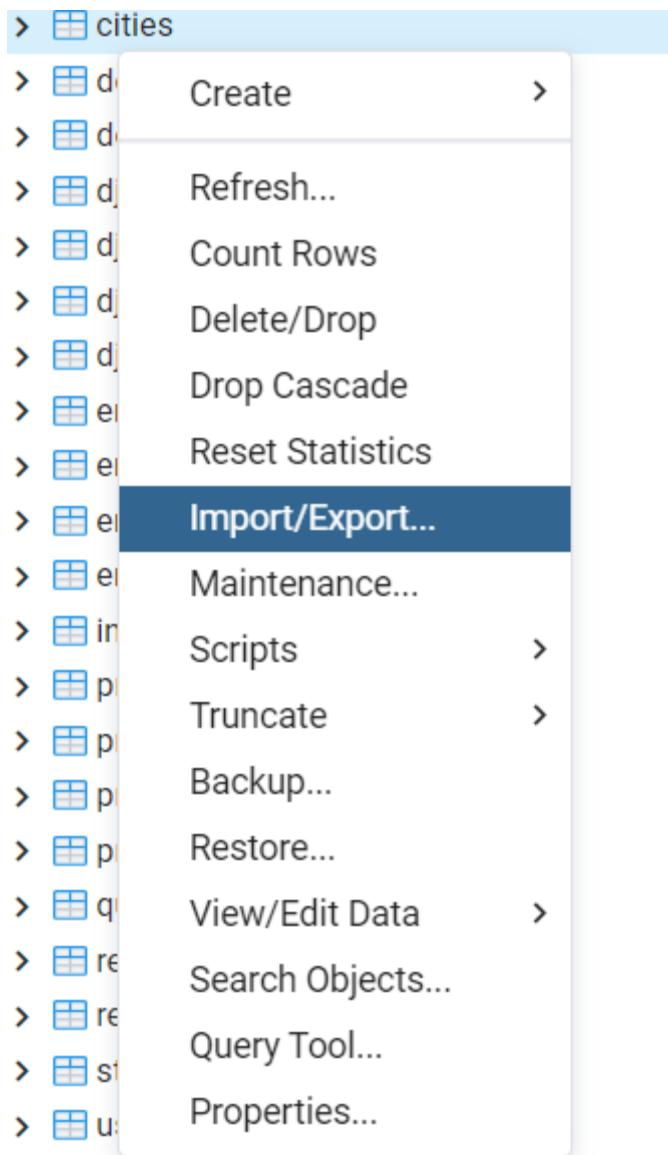
Birmingham,GBR,1111300,3683000,1111300

Bogota,COL,7878780,9800000,7878780

Brasilia,BRA,2556150,3919860,2556150

Brazzaville,CON,1027000,0,1027000

<



Import/Export data - table 'cities'

Options Columns

Import/Export

Import

File Info

Filename

 ...

Format

CSV

Encoding

Select an item...

Miscellaneous

OID

No

Header

No

Delimiter

Select from list...

Specifies the character that separates columns within each row (line) of the file. The default is a tab character in text format, a comma in CSV format. This



Please provide filename



Cancel

OK



Options Columns

Import/Export

Import

File Info

Filename

D:\cities.csv



Format

csv



Encoding

UTF8



Miscellaneous

OID

No

Header

No

Delimiter

,



Specifies the character that separates columns within each row (line) of the file. The default is a tab character in text format, a comma in CSV format. This must be a single one-byte character. This option is not allowed when using binary format.

Cancel

OK



test/postgres@PostgreSQL 13 ▾

Query Editor

```
1 create table cities(name character varying, country_code character varying,  
2                      city_proper_pop character varying, metro_pop character varying,  
3                      urbanarea_pop character varying)
```

Data Output Explain Messages Notifications Query History

CREATE TABLE

Query returned successfully in 72 msec.

Import - Copying table data

Copying table data 'public.cities' on database 'test' and server
(localhost:5432)

Sun Mar 07 2021 11:49:37 GMT+0530 (India Standard Time)

⌚ 0.14 seconds

ⓘ More details...

✖ Stop Process



Successfully completed.

Importing table should have column with character varying irrespective of data to import successfully FTS system data to DB

SQL LIKE

Like keyword is case sensitive

Fetch cities whose name have atleast 'A' once

```
1 select * from cities WHERE name like '%A%'
```

Data Output

Explain

Messages

Notifications

Query History

	name character varying	country_code character varying	city_proper_pop character varying	metro_pop character varying	urbanarea_pop character varying
1	Abidjan	CIV	4765000	0	4765000
2	Abu Dhabi	ARE	1145000	0	1145000
3	Abuja	NGA	1235880	6000000	1235880
4	Accra	GHA	2070460	4010050	2070460
5	Addis Ababa	ETH	3103670	4567860	3103670
6	Ahmedabad	IND	5570580	0	5570580
7	Alexandria	EGY	4616620	0	4616620
8	Algiers	DZA	3415810	5000000	3415810
9	Almaty	KAZ	1703480	0	1703480
10	Ankara	TUR	5271000	4585000	5271000
11	Auckland	NZL	1495000	1614300	1495000
12	Buenos Aires	ARG	3054300	14122000	3054300
13	Abidjan	CIV	4765000	4765000	4765000
14	Abu Dhabi	ARE	1145000	0	1145000
15	Abuja	NGA	1235880	6000000	1235880

✓ Successful

Query Editor

```
1 select * from cities WHERE name like '%a%'
```

Data Output Explain Messages Notifications Query History

	name character varying	country_code character varying	city_proper_pop character varying	metro_pop character varying	urbanarea_pop character varying
106	Brazzaville	COG	1827000	0	1827000
107	Brisbane	AUS	1180280	2349700	1180280
108	Bucharest	ROM	1883420	2272160	1883420
109	Budapest	HUN	1759410	2927940	1759410
110	Busan	KOR	3510830	8202240	3510830
111	Cairo	EGY	10230400	18290000	10230400
112	Calgary	CAN	1235170	1214840	1235170
113	Cali	COL	2400650	3400000	2400650
114	Caloocan	PHL	1583980	0	1583980
115	Campinas	BRA	1164100	3094180	1164100
116	Cape Town	ZAF	3740030	0	3740030
117	Caracas	VEN	1943900	2923960	1943900
118	Casablanca	MAR	5117830	6861740	5117830
119	Changchun	CHN	3815270	7674440	3815270
120	Changsha	CHN	7044120	0	7044120

Lower function

```
1 select lower(name) from cities WHERE lower(name) like '%a%'
```

Data Output Explain Messages Notifications Query History

lower
text 

1	abidjan
2	abu dhabi
3	abuja
4	accra
5	addis ab...
6	ahmedab...
7	alexandria
8	algiers
9	almaty
10	ankara
11	auckland
12	baghdad
13	baku
14	bandung
15	bangkok
16	-----

Query Editor

```
1 select UPPER(name) from cities WHERE UPPER(name) like '%A%'
```

Data Output Explain Messages Notifications Query History

	upper text	lock
106	BIRMINGHAM	
107	BOGOTA	
108	BRASILIA	
109	BRAZZAVILLE	
110	BRISBANE	
111	BUCHAREST	
112	BUDAPEST	
113	BUENOS AIRES	
114	BUSAN	
115	CAIRO	
116	CALGARY	
117	CALI	
118	CALOOCAN	
119	CAMPINAS	
120	CAPETOWN	

Query Editor

```
1 select name from cities WHERE name ilike '%a'
```

Data Output

Explain

Messages

Notifications

Query History

	name character varying	lock
1	Abuja	
2	Accra	
3	Addis Ababa	
4	Alexandria	
5	Ankara	
6	Barcelona	
7	Barranquilla	
8	Basra	
9	Bogota	
10	Brasilia	
11	Casablanca	
12	Changsha	
13	Cordoba	
14	Curitiba	
15	Dhaka	
16	Doha	

```
ALTER TABLE employee_constraint DROP COLUMN sex;
```

```
SELECT * FROM employee_constraint
```

```
SELECT * FROM demo.employee_dummy  
insert into demo.employee_dummy(name,mobile_number) values('John',0707070607);
```

"ALTER TABLE "table_name" ADD "column_name" DATA TYPE"

```
ALTER TABLE employee_constraint ADD gender char;
```

```
SELECT * FROM employee_constraint
```

```
ALTER TABLE employee_constraint ADD country character varying(20);
```

```
UPDATE EMPLOYEE_constraint set gender='M' where id in (2,6)
```

```
ALTER TABLE employee_constraint ALTER column gender TYPE character varying(10),  
ALTER country TYPE character varying(50);
```

```
ALTER TABLE employee_constraint RENAME COLUMN gender to sex;
```

```
DROP TABLE demo.employee_dummy;  
DELETE FROM demo.employee_dummy where cond1=? ,....,Cond n=?;  
TRUNCATE TABLE demo.employee_dummy ;
```

- DELETE (conditional delete) & DELETE query takes where condition
- DROP deletes whole data of the table along with its structure
- TRUNCATE delete only data of the table

```
8  
9  insert into employee_demo1(first_name,last_name,gender,mobile)  
10 values('john','smith','m',99877055545),  
11 ('jacob','smith','m',9987789978) ,  
12 ('scott','smith','m',9987700078);  
13  
14 delete from employee_demo1 where id=2  
15  
16 DROP table employee_demo1
```

Data Output Explain Messages Notifications Query History

ERROR: relation "employee_demo1" does not exist

LINE 1: SELECT * from employee_demo1
^

SQL state: 42P01

Character: 15

--single column

SELECT "Columnname" from "table_name"

--multi

SELECT "Columnname1","Columnname1"....."Columnnamen" from "table_name"

--All

SELECT * from "tablename";

```
delete from demo.employee_dummy

CREATE TABLE demo.employee_dummy
(
    id serial,
    name character varying(100) COLLATE pg_catalog."default" NOT NULL,
    mobile_number bigint NOT NULL
)
```

-- No duplicate

```
SELECT DISTINCT name, mobile_number FROM demo.employee
```

```
2           last_name character varying(50) ,gender char, mob
3   SELECT DISTINCT first_name from
4   employee_demo1
5
6   insert into employee_demo1(first_name, last_name, gender, mobile) values(
7
8
9   insert into employee_demo1(first_name, last_name, gender, mobile)
L0   values('john','smith','m',99877055545),
L1   ('jacob','smith','m',9987789978) ,
L2   ('scott','smith','m',9987700078);
L3
```

Data Output Explain Messages Notifications Query History

	first_name	
	character varying (100)	🔒
1	jacob	
2	john	
3	scott	

-- No duplicate
SELECT DISTINCT name,mobile_number FROM demo.employee
WHERE mobile_number>23

```
CREATE TABLE demo.employee_dummy
(
    id serial,
    name character varying(100) COLLATE pg_catalog."default"
NOT NULL,
    mobile_number bigint NOT NULL,
    CONSTRAINT employee_pkeyp PRIMARY KEY (id)
)
TABLESPACE pg_default;

ALTER TABLE demo.employee
OWNER to postgres;
```

```
CREATE TABLE IF NOT EXISTS demo.employee_dummy1  
(  
    id serial,  
    name character varying(100) COLLATE pg_catalog."default"  
NOT NULL,  
    mobile_number bigint NOT NULL  
)
```

```
INSERT INTO demo.employee(  
    name, mobile_number)  
VALUES ( 'Tej', 23);
```

```
select * from demo.employee where id = 3 or id=2;
```

```
SELECT * FROM demo.employee WHERE name='Bhanu' and  
id=20
```

```
SELECT * FROM demo.employee --limit 2
```

```
SELECT * FROM demo.employee order by mobile_number ASC
```

```
SELECT * FROM demo.employee order by mobile_number DESC
```

```
select length(name),name from demo.employee
```

Data type

character varying

numeric

bigint

integer

char

boolean

6

test/postgres@PostgreSQL 13 ▾

Query Editor Query History

Multiple rows insert at once in SQL

```
Query Editor  Query History
10 CREATE TABLE users_education_dtls(id serial,
11                                     user_id integer REFERENCES users(id),
12                                     qualification character varying,
13                                     college character varying);
14
15
16 INSERT INTO users(first_name,gender) values('Teja','M');
17
18 INSERT INTO users_education_dtls(user_id,qualification,college) values
19 (1,'PUC','BMS'),
20 (1,'BTECH','BMS'),
21 (1,'MTECH','PES');
22
```

[Query Editor](#) [Query History](#)

```
8 select * from users
9 INSERT INTO users(first_name,gender) values('Teja','M');
10
11 CREATE TABLE users_education_dtls(id serial,
12                                     user_id integer REFERENCES users(id),
13                                     qualification character varying,
14                                     college character varying);
15
16 INSERT INTO users_education_dtls(user_id,qualification,college) values
17 (1,'PUC','BMS'),
18 (1,'BTECH','BMS'),
19 (1,'MTECH','PES');
20
21 select * from users_education_dtls
22
23 DELETE from users where id=1
```

[Data Output](#) [Explain](#) [Messages](#) [Notifications](#)

ERROR: update or delete on table "users" violates foreign key constraint "users_education_dtls_user_id_fkey" on table "users_education_dtls"
DETAIL: Key (id)=(1) is still referenced from table "users_education_dtls".
SQL state: 23503

UPDATE Statement in SQL

Data Output							Explain	Messages	Notifications	Query History
	id integer	first_name character varying (100)	last_name character varying (50)	gender character (1)	mobile bigint					
1	1	john	smith	m	9987789678					
2	2	john	smith	m	9877055545					
3	3	jacob	smith	m	9987789978					
4	4	scott	smith	m	9987700078					

Syntax : UPDATE TABLE_NAME SET col1=new value,...coln=newvalue [where Condition]

Ex : UPDATE EMPLOYEE_DEMO1 SET first_name='Harshitha' where ID=2

UPDATE Statement in SQL

```
1/  
18 UPDATE EMPLOYEE_DEMO1 SET first_name='Harshitha' where ID=2  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28
```

Data Output Explain Messages Notifications Query History

UPDATE 1

Query returned successfully in 70 msec.

```
18 UPDATE EMPLOYEE_DEMO1 SET first_name='Harshitha' where ID=2  
19  
20 SELECT * FROM EMPLOYEE_DEMO1 ORDER BY ID  
21  
22  
23  
24  
25  
26  
27  
28
```

Data Output Explain Messages Notifications Query History

	id integer	first_name character varying (100)	last_name character varying (50)	gender character (1)	mobile bigint
1	1	john	smith	m	987789678
2	2	Harshitha	smith	m	9877055545
3	3	jacob	smith	m	987789978
4	4	scott	smith	m	987700078

ADD COLUMN in SQL

Syntax : ALTER TABLE “table_name” ADD COLUMN1 datatype;

```
46  
47 ALTER TABLE employee_demo1 ADD email character varying(50);  
48  
49  
50  
51  
52  
53 UPDATE employee_demo1 SET EMAIL = first_name||last_name||'@xyz.com'
```

Data Output Explain Messages Notifications Query History

	id integer	first_name character varying (100)	last_name character varying (50)	gender character (1)	mobile bigint	
1		john	smith	m	987789678	
2		Harshitha	smith	m	9877055545	
3		jacob	smith	m	987789978	
4		scott	smith	m	987700078	

```
45  
46  
47 ALTER TABLE employee_demo1 ADD email character varying(50);  
48  
49 SELECT * FROM EMPLOYEE_DEMO1 ORDER BY ID  
50  
51
```

Data Output Explain Messages Notifications Query History

	id integer	first_name character varying (100)	last_name character varying (50)	gender character (1)	mobile bigint	email character varying (50)
1	1	john	smith	m	987789678	[null]
2	2	Harshitha	smith	m	9877055545	[null]
3	3	jacob	smith	m	987789978	[null]
4	4	scott	smith	m	987700078	[null]

```
8  
9  SELECT * FROM EMPLOYEE_DEMO1 ORDER BY ID  
0  
1  
2  
3  
4  UPDATE employee_demo1 SET EMAIL = first_name||last_name||'@xyz.com'  
5  
6  
7
```

Data Output Explain Messages Notifications Query History

	id integer	first_name character varying (100)	last_name character varying (50)	gender character (1)	mobile bigint	email character varying (50)	
	1	john	smith	m	9987789678	johnsmith@xyz.com	
	2	Harshitha	smith	m	9877055545	Harshithasmith@xyz.com	
	3	jacob	smith	m	9987789978	jacobsmith@xyz.com	
	4	scott	smith	m	9987700078	scottsmith@xyz.com	

```
ALTER TABLE employee_demo1 ADD email character varying(50);
```

```
SELECT * FROM EMPLOYEE_DEMO1 ORDER BY ID
```

```
UPDATE employee_demo1 SET EMAIL = first_name||last_name||'@xyz.com'
```

a Output Explain Messages Notifications Query History

:OR: column "email" of relation "employee_demo1" already exists
. state: 42701

```
ALTER TABLE employee_demo1 ADD email character  
varying(50);
```

DROP COLUMN in SQL

```
21  
22 ALTER TABLE EMPLOYEE_DEMO1 DROP COLUMN email;  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32
```

Data Output Explain Messages Notifications Query History

	id integer	first_name character varying (100)	last_name character varying (50)	gender character (1)	mobile bigint	email character varying (50)	
1	1	john	smith	m	987789678	johnsmith@xyz.com	
2	2	Harshitha	smith	m	9877055545	Harshithasmith@xyz.com	
3	3	jacob	smith	m	987789978	jacobsmith@xyz.com	
4	4	scott	smith	m	987700078	scottsmith@xyz.com	

Create table without constraints

```
1 create table product_demo(id serial,  
2                         product_name character varying,  
3                         expiry_date date ,  
4                         product_cost numeric );  
5  
6 insert into product_demo(product_name,expiry_date,product_cost)  
7 values(null,'01-01-2021',888.90)
```

Data Output Explain Messages Notifications Query History

INSERT 0 1

Query returned successfully in 87 msec.

```
create table product_demo(id serial,  
                           product_name character varying,  
                           expiry_date date ,  
                           product_cost numeric );
```

```
insert into product_demo(product_name,expiry_date,product_cost)  
values(null,'01-01-2021',888.90)
```

SQL Constraints

1. NOT NULL Constraints
2. DEFAULT Constraints
3. UNIQUE Constraints
4. CHECK Constraints

Query Editor Query History

```
1 CREATE TABLE besant.employees(id serial,
2                               first_name character varying(50) NOT NULL,
3                               second_name character varying(20) NOT NULL,
4                               dob date,
5                               mobile_no bigint UNIQUE NOT NULL,
6                               email_id character varying(50) UNIQUE );
7
8
9 INSERT INTO besant.employees(first_name,dob,mobile_no,email_id) values('Scott','11-02-1989',9899898898,'scott@gmail.com');
10
11
12
13
```

Data Output Explain Messages Notifications

ERROR: null value in column "second_name" of relation "employees" violates not-null constraint
DETAIL: Failing row contains (1, Scott, null, 1989-02-11, 9899898898, scott@gmail.com).
SQL state: 23502

DROP CONSTRAINT ON A COLUMN

```
ALTER TABLE product_demo  
ALTER column product_cost DROP not null;
```

ADD CONSTRAINT ON AN Existing COLUMN

```
ALTER TABLE public.product_ingredients  
ALTER COLUMN product_id SET NOT NULL;
```

ADD CONSTRAINT ON A COLUMN

```
ALTER TABLE product_demo  
ALTER column product_cost set  
not null;
```

round function in sql

```
6  
7 select avg(sales) from retail_store  
8  
9 select round(avg(sales)) from retail_store  
10  
11  
12  
13
```

Data Output		Explain	Messages	Notifications	Query History
avg	numeric				
1	697.2222222222222222	🔒			

Data Output		Explain	Messages	Notifications	Query History
round	numeric	🔒			
1	697				

DEFAULT Constraint

```
--  
40 create table courses(id serial, course_name character varying ,price integer default 10000);  
41  
42 insert into courses(course_name) values('java'),('sql');  
43  
44 insert into courses(course_name,price) values('dotnet',1000),('c',500);  
45
```

Data Output Explain Messages Notifications Scratch Pad

	id integer	course_name character varying	price integer
1	1	java	10000
2	2	sql	10000
3	3	dotnet	1000
4	4	c	500

UNIQUE Constraint

```
41 create table courses(id serial, course_name character varying UNIQUE ,price integer default 10000);  
42  
43 insert into courses(course_name) values('java'),('sql');  
44  
45 insert into courses(course_name,price) values('java',1000),('c',500);  
46  
47
```

Data Output Explain Messages Notifications Scratch Pad

ERROR: duplicate key value violates unique constraint "courses_course_name_key"
DETAIL: Key (course_name)=(java) already exists.
SQL state: 23505

CHECK Constraint

```
41 create table courses(id serial, course_name character varying UNIQUE ,price integer default 10000  
42           ,CHECK (price>=5000));  
43  
44 insert into courses(course_name) values('java'),('sql');  
45  
46 insert into courses(course_name,price) values('java',1000),('c',500);  
47
```

Data Output Explain Messages Notifications Scratch Pad

ERROR: new row for relation "courses" violates check constraint "courses_price_check"
DETAIL: Failing row contains (3, java, 1000).
SQL state: 23514

CASE statements

- Contains a WHEN , THEN , and ELSE statement, finished with
END

```
CASE WHEN x = 1 THEN 'a'  
      WHEN x = 2 THEN 'b'  
      ELSE 'c' END AS new_column
```

Welcome to the course!

JOINING DATA IN SQL

A dark blue circular icon containing the letters "SQL" in white.

left_table

id	val
1	L1
2	L2
3	L3
4	L4

right_table

id	val
1	R1
4	R2
5	R3
6	R4

prime_ministers table

country	continent	prime_minister
Egypt	Africa	Sherif Ismail
Portugal	Europe	Antonio Costa
Vietnam	Asia	Nguyen Xuan Phuc
Haiti	North America	Jack Guy Lafontant
India	Asia	Narendra Modi
Australia	Oceania	Malcolm Turnbull
Norway	Europe	Erna Solberg
Brunei	Asia	Hassanal Bolkiah
Oman	Asia	Qaboos bin Said al Said
Spain	Europe	Mariano Rajoy

presidents table

```
SELECT *
FROM presidents;
```

country	continent	president
Egypt	Africa	Abdel Fattah el-Sisi
Portugal	Europe	Marcelo Rebelo de Sousa
Haiti	North America	Jovenel Moise
Uruguay	South America	Jose Mujica
Liberia	Africa	Ellen Johnson Sirleaf
Chile	South America	Michelle Bachelet
Vietnam	Asia	Tran Dai Quang

INNER JOIN in SQL

```
SELECT p1.country, p1.continent,  
       prime_minister, president  
FROM prime_ministers AS p1  
INNER JOIN presidents AS p2  
ON p1.country = p2.country;
```

country	continent	prime_minister	president
Egypt	Africa	Sherif Ismail	Abdel Fattah el-Sisi
Portugal	Europe	Antonio Costa	Marcelo Rebelo de Sousa
Vietnam	Asia	Nguyen Xuan Phuc	Tran Dai Quang
Haiti	North America	Jack Guy Lafontant	Jovenel Moise

```
create table products(id serial primary key,product_name character varying not null,price numeric(5,2));
```

```
create table public.vendors(id serial,vendor_name character varying(100) not null,  
                           product_id integer REFERENCES public.products(id));
```

```
insert into vendors(vendor_name,product_id) values('RG',1);  
insert into vendors(vendor_name,product_id) values('BG',3);  
insert into vendors(vendor_name,product_id) values('SS Vendor',2);  
insert into vendors(vendor_name,product_id) values('SS Vendor',3);  
insert into vendors(vendor_name,product_id) values('SS Vendor',4);  
insert into vendors(vendor_name,product_id) values('SS Vendor',5);
```

```
insert into vendors(vendor_name,product_id) values('Philp logistics',2);  
insert into vendors(vendor_name,product_id) values('Philp logistics',1);  
insert into vendors(vendor_name,product_id) values('Philp logistics',5);
```

```
insert into vendors(vendor_name,product_id) values('Kumar Vendor',4);  
insert into vendors(vendor_name,product_id) values('Kumar Vendor',5);
```

```
41      select * from products as p ,vendors as v where p.id=v.product_id  
42  
43  
44  
45
```

Data Output Explain Messages Notifications Query History

	id integer	product_name character varying	price numeric (5,2)	id integer	vendor_name character varying (100)	product_id integer	
1	2	pepsi	10.00	1	Philp logistics	2	
2	1	coke	10.00	2	Philp logistics	1	
3	5	fanta	10.00	3	Philp logistics	5	
4	1	coke	10.00	5	RG	1	
5	3	maaza	10.00	6	BG	3	
6	2	pepsi	10.00	7	SS Vendor	2	
7	3	maaza	10.00	8	SS Vendor	3	
8	4	mountain dew	10.00	9	SS Vendor	4	
9	5	fanta	10.00	10	SS Vendor	5	

```
51
52     select p.product_name,v.vendor_name  from products as p ,vendors as v where p.id=v.product_id
53
```

Data Output Explain Messages Notifications Query History

	product_name character varying	vendor_name character varying (100)
1	pepsi	Philp logistics
2	coke	Philp logistics
3	fanta	Philp logistics
4	coke	RG
5	maaza	BG
6	pepsi	SS Vendor
7	maaza	SS Vendor
8	mountain dew	SS Vendor
9	fanta	SS Vendor

```
43      select *  from products as p  join vendors v on v.product_id = p.id
44
45
```

Data Output Explain Messages Notifications Query History

	id integer	 product_name character varying	 price numeric (5,2)	 id integer	 vendor_name character varying (100)	 product_id integer	
1	2	pepsi	10.00	1	Philp logistics	2	
2	1	coke	10.00	2	Philp logistics	1	
3	5	fanta	10.00	3	Philp logistics	5	
4	1	coke	10.00	5	RG	1	
5	3	maaza	10.00	6	BG	3	
6	2	pepsi	10.00	7	SS Vendor	2	
7	3	maaza	10.00	8	SS Vendor	3	
8	4	mountain dew	10.00	9	SS Vendor	4	
9	5	fanta	10.00	10	SS Vendor	5	
10	5	fanta	10.00	11	high Vendor	5	

```
42  
43     select *      from products as p right join vendors v on v.product_id = p.id  
44  
45
```

Data Output Explain Messages Notifications Query History

	id integer	product_name character varying	price numeric (5,2)	id integer	vendor_name character varying (100)	product_id integer	
1	2	pepsi	10.00	1	Philp logistics	2	
2	1	coke	10.00	2	Philp logistics	1	
3	5	fanta	10.00	3	Philp logistics	5	
4	1	coke	10.00	5	RG	1	
5	3	maaza	10.00	6	BG	3	
6	2	pepsi	10.00	7	SS Vendor	2	
7	3	maaza	10.00	8	SS Vendor	3	
8	4	mountain dew	10.00	9	SS Vendor	4	
9	5	fanta	10.00	10	SS Vendor	5	
10	5	fanta	10.00	11	high Vendor	5	
11	[null]	[null]	[null]	12	one Vendor	[null]	

```
42  
43     select *    from products as p left join vendors v on v.product_id = p.id  
44  
45
```

Data Output Explain Messages Notifications Query History

	id integer	 product_name character varying	 price numeric (5,2)	 id integer	 vendor_name character varying (100)	 product_id integer	
1	2	pepsi	10.00	1	Philp logistics	2	
2	1	coke	10.00	2	Philp logistics	1	
3	5	fanta	10.00	3	Philp logistics	5	
4	1	coke	10.00	5	RG	1	
5	3	maaza	10.00	6	BG	3	
6	2	pepsi	10.00	7	SS Vendor	2	
7	3	maaza	10.00	8	SS Vendor	3	
8	4	mountain dew	10.00	9	SS Vendor	4	
9	5	fanta	10.00	10	SS Vendor	5	
10	5	fanta	10.00	11	high Vendor	5	
11	6	7up	10.00	[null]	[null]	[null]	

```
select * from public.products p , public.vendors v where  
p.id=v.product_id  
AND vendor_name IN ('RG','BG')
```

Query Editor

```
1 select * from public.products p , public.vendors v where p.id=v.product_id  
2 AND vendor_name IN ('RG','BG')
```

Data Output Explain Messages Notifications Query History

	id integer	product_name character varying	price numeric (5,2)	id integer	vendor_name character varying (100)	product_id integer	
1	1	coke	10.00	5	RG	1	
2	3	maaza	10.00	6	BG	3	

SUB- QUERIES

WHERE are the
subqueries?

INTERMEDIATE SQL

SQL

What is a subquery?

- A query *nested* inside another query

```
SELECT column  
FROM (SELECT column  
      FROM table) AS subquery;
```

What do you do with subqueries?

- Can be in *any* part of a query
 - `SELECT` , `FROM` , `WHERE` , `GROUP BY`
- Can return a variety of information
 - Scalar quantities (`3.14159` , `-2` , `0.001`)
 - A list (`id = (12, 25, 392, 401, 939)`)
 - A table

fetch all vendors whose product price is greater than 500

```
43 select * from vendors where product_id in  
44 (select id from products where price>500)  
45  
46  
47
```

Data Output Explain Messages Notifications Scratch Pad

	id integer	vendor_name character varying (100)	product_id integer	
1		8 BG		3
2		10 SS Vendor		3
3		11 SS Vendor		4
4		12 SS Vendor		5
5		15 Philp logistics		5
6		16 Kumar Vendor		4
7		17 Kumar Vendor		5

Fetch all product details whose vendor name contains letter ‘h’

```
40 select * from products where id in (
41 select distinct product_id from vendors where vendor_name ilike '%H%')
42
43
44
45
46
47
48
```

Data Output Explain Messages Notifications Scratch Pad

	id [PK] integer	product_name character varying	price numeric (5,2)
1	1	Hyundai-i20	100.00
2	2	Hyundai-venue	200.00
3	5	Honda-City	900.00

like and ilike keywords are used to
search the data in the table

- Like is case sensitive
- Ilike is not case sensitive

Fetch all vendor details whose
Name starts with 's'

Second letter starts with 's'

```
54 select * from vendors where vendor_name like 's%'  
55  
56 |  
57  
58
```

Data Output Explain Messages Notifications Scratch Pad

	id integer	vendor_name character varying (100)	product_id integer

✓ Successfully run. Total query time: 0.000 ms.

Last letter ends with 'r'

```
53  
54 select * from vendors where vendor_name ilike '%_r'  
55  
56  
57  
58
```

Data Output

Explain

Messages

Notifications

Scratch Pad

	<u>id</u> integer	 <u>vendor_name</u> character varying (100) 	<u>product_id</u> integer 
1	9	SS Vendor	2
2	10	SS Vendor	3
3	11	SS Vendor	4
4	12	SS Vendor	5
5	16	Kumar Vendor	4
6	17	Kumar Vendor	5

```
54 select * from vendors where vendor_name ilike '_s%'  
55  
56  
57  
58
```

Data Output

Explain

Messages

Notifications

Scratch Pad

	id integer	 	vendor_name character varying (100)	 	product_id integer	
1	9		SS Vendor		2	
2	10		SS Vendor		3	
3	11		SS Vendor		4	
4	12		SS Vendor		5	

Timestamp vs Date vs now()

How to check current timestamp ?

```
18 select current_timestamp;  
19  
20  
21  
22  
23
```

Data Output Explain Messages Notifications Scratch Pad

	current_timestamp	timestamp with time zone	🔒
1	2021-09-26 12:22:22.733286+05:30		

```
19  
20 select now()  
21  
22  
23
```

Data Output Explain Messages Notifications

	now	timestamp with time zone	🔒
1	2021-09-26 12:23:18.44276+05:30		

```
27 select age('2012-01-01','2021-09-26')
28 |
29
30
```

Data Output Explain Messages Notifications Scratch Pad

	age interval	🔒
1	-9 years -8 mons -25 days	

```
24 select date_part('quarter',Timestamp '2017-01-01')  
25  
26  
27  
28 |  
29  
30
```

Data Output Explain Messages Notifications Scratch Pad

	date_part	double precision
1		1

```
24 select date_part('year',Timestamp '2017-01-01')  
25  
26  
27  
28  
29  
30
```

Data Output Explain Messages Notifications Scratch Pad

	date_part	double precision
1		2017

```
24 select date_part('century',Timestamp '2017-01-01')
25
26
27
28
```

Data Output Explain Messages Notifications Scratch Pad

	date_part	lock
	double precision	
1	21	

```
--  
24 select date_part('month',Timestamp '2017-01-01')  
25  
26  
27  
28
```

Data Output Explain Messages Notifications Scratch Pad

	date_part	lock
	double precision	
1	1	

```
24 select date_part('week',Timestamp '2021-09-26')  
25  
26  
27  
28
```

Data Output Explain Messages Notifications Scratch Pad

	date_part	lock
	double precision	
1	38	

```
24 select EXTRACT(day from updated_date) from staff  
25  
26  
27  
28
```

Data Output Explain Messages Notifications Scratch Pad

	date_part	double precision
12		26
13		26
14		26
15		26
16		26
17		26

```
24 select EXTRACT(month from updated_date) from staff  
25  
26  
27  
28
```

Data Output

Explain

Messages

Notifications

Scratch Pad

	date_part	lock
7		9
8		9
9		9
10		9
11		9
12		9
13		9

Can I convert string to date ?

```
24 select to_date('10 FEB 2021','dd mon yyyy') from staff  
25  
26  
27  
28
```

Data Output

Explain

Messages

Notifications

Scratch Pad

	to_date date	🔒
1	2021-02-10	
2	2021-02-10	
3	2021-02-10	
4	2021-02-10	
5	2021-02-10	
6	2021-02-10	
7	2021-02-10	

```
select to_date('20210926','YYYYMMDD') from staff
```

:a Output

Explain

Messages

Notifications

Scratch Pad

to_date	date
	2021-09-26

Complete the code to return the output

```
SELECT title, release_year  
FROM films  
? release_year BETWEEN 2002 AND 2005  
LIMIT 5;
```

title	release_year
25th Hour	2002
28 Days Later...	2002
40 Days and 40 Nights	2002
8 Mile	2002
8 Women	2002

Fill in the blanks

HAVING 1 WHERE 2 WHEN 3 CASE 4 IF 5

Select the code to return the output

name	employee_number	founding_year
Root	1792	2012
Software	1944	2011

Select the code

```
SELECT name, employee_number, founding_year  
FROM companies  
WHERE founding_year BETWEEN 2010 AND 2013  
AND employee_number BETWEEN 200 AND 250  
LIMIT 2;
```

PRESS 1

```
SELECT name, employee_number, founding_year  
FROM companies  
WHERE founding_year BETWEEN 2012 AND 2013  
AND employee_number > 300  
LIMIT 2;
```

PRESS 2

```
SELECT name, employee_number, founding_year  
FROM companies  
WHERE founding_year BETWEEN 2010 AND 2013  
AND employee_number > 300  
LIMIT 2;
```

PRESS 3

What is important in ORDER BY clause for sorting multiple columns?

Select the correct answer

The number of columns

PRESS **1**

The order of columns

PRESS **2**

The types of columns

PRESS **3**

What does the `LIMIT` keyword do?

Select the correct answer

- It limits the number of rows returned.

PRESS **1**

- It limits the number of columns returned.

PRESS **2**

- It limits the number of rows and columns returned.

PRESS **3**

Complete the code to return the output

```
SELECT founding_year, COUNT(*)  
FROM companies  
? BY founding_year;
```

foundng_year	count
2005	2
2015	4
1995	1
2012	2
2011	3
2014	1
2004	1
2009	1
2013	4
2008	1

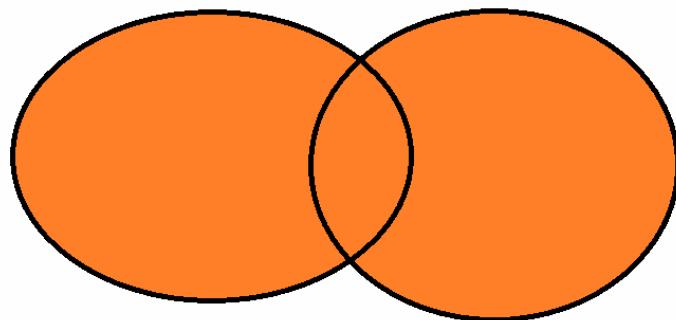
Fill in the blanks

GROUP 1

SUM 2

ORDER 3

Union in SQL



each UNION query must have the same number of columns

UNION types character varying and integer cannot be matched

```
15  
16 select company_name from most_trusted_companies  
17 union  
18 select id from top_companies;
```

Data Output Explain Messages Notifications Scratch Pad

ERROR: UNION types character varying and integer cannot be matched

LINE 3: select id from top_companies;
 ^

SQL state: 42804

Character: 63

Union won't give us duplicate data

```
7  
8 select * from top_companies;  
9  
10
```

Data Output Explain Messages Notifications Scratch

	id integer	company_name character varying	established integer
1	1	TATA	1951
2	2	Reliance	1951
3	3	Apollo	1951
4	4	Microsoft	1951

```
--  
12 select * from most_trusted_companies;
```

```
13  
14  
15  
16
```

Data Output Explain Messages Notifications ⚙

	id integer	company_name character varying	established integer
1	1	TATA	1951
2	2	Reliance	1970
3	3	Medplus	2012
4	4	Google	1980

```
--  
23 select company_name from most_trusted_companies  
24 union  
25 select company_name from top_companies;
```

Data Output Explain Messages Notifications Scratch Pad

	company_name
1	TATA
2	Reliance
3	Apollo
4	Microsoft
5	Google
6	Medplus

✓ Successfully run. T

We can use the ORDER BY clause
in the second Query

Guess the output

```
23 select company_name from most_trusted_companies order by company_name  
24 union  
25 select company_name from top_companies
```

```
22  
23 select company_name from most_trusted_companies  
24 union  
25 select company_name from top_companies order by company_name
```

Data Output Explain Messages Notifications Scratch Pad

	company_name
1	Apollo
2	Google
3	Medplus
4	Microsoft
5	Reliance
6	TATA

UNION ALL

```
22  
23 select company_name from most_trusted_companies  
24 union all  
25 select company_name as my_comp from top_companies order by company_name
```

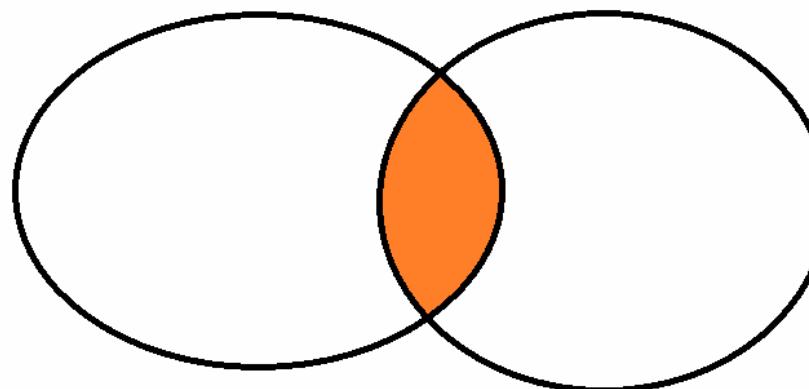
Data Output Explain Messages Notifications Scratch Pad

	company_name
1	Apollo
2	Google
3	Medplus
4	Microsoft
5	Reliance
6	Reliance
7	TATA
8	TATA

Union : will not allow the duplicate values

Union all: Will allow duplicate values

POSTGRESQL INTERSECT

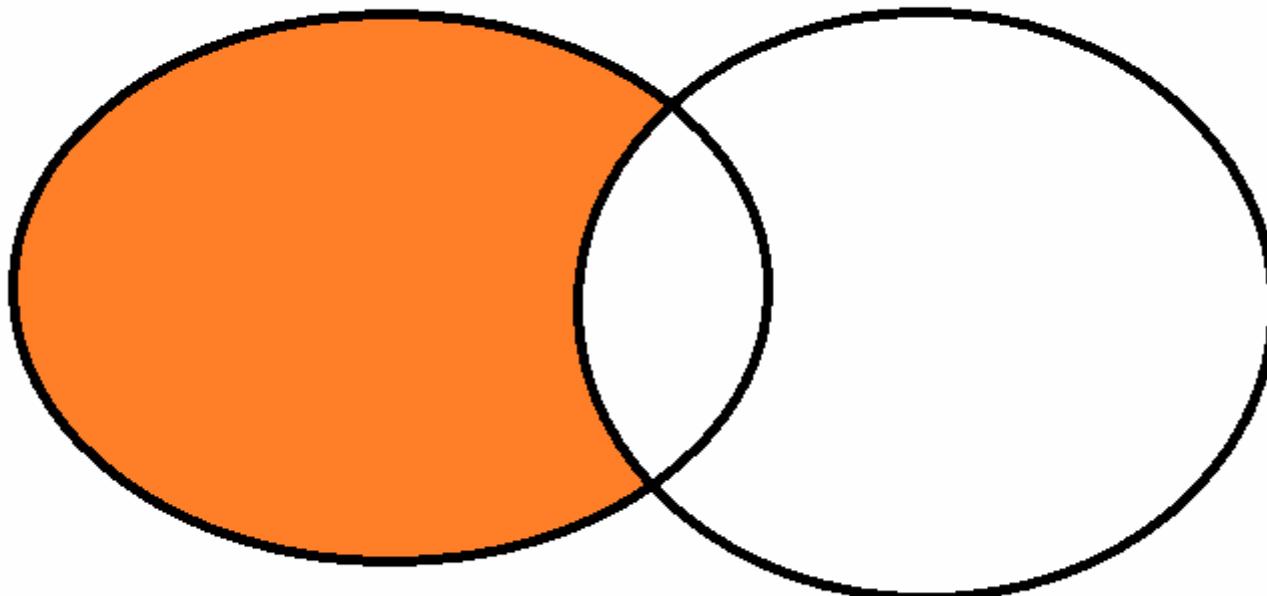


```
22  
23 select company_name from most_trusted_companies  
24 INTERSECT  
25 select company_name as my_comp from top_companies order by company_name
```

Data Output Explain Messages Notifications Scratch Pad

	company_name
1	Reliance
2	TATA

POSTGRESQL EXCEPT



Except operator will return
all records from initial dataset and
then eliminate all values
from second dataset

```
23 select company_name from most_trusted_companies
24 EXCEPT
25 select company_name as my_comp from top_companies order by company_name
```

Data Output Explain Messages Notifications Scratch Pad

	company_name
1	Google
2	Medplus

```
36  
37 select * from most_trusted_companies FETCH FIRST 5 rows only  
38  
39  
40
```

Data Output Explain Messages Notifications Scratch Pad

	id integer 	company_name character varying 	established integer 	
1	1	TATA	1951	
2	2	Reliance	1970	
3	3	Medplus	2012	
4	4	Google	1980	
5	5	Wipro	1951	

```
36  
37 select * from most_trusted_companies limit 10 offset 5  
38  
39  
40
```

Data Output Explain Messages Notifications Scratch Pad

	id integer	company_name character varying	established integer
1	6	Infosys	1970
2	7	IBM	2012
3	8	ST micro	1980
4	9	Cognizant	1951
5	10	Dell	1970
6	11	Lenovo	2012
7	12	HP	1980

✓ Successfully run. Total query runti

Query Editor Query History

```
1 SELECT count(*) FROM public.testproduct;
2
3 select count(1) from public.testproduct ;
4
5
6
```

```
1
2 select * from products
3
4 select string_agg(name,',') , price from products group by price
```

Data Output Explain Messages Notifications Scratch Pad

	string_agg
	text
1	Vento-VW
2	iphone13
3	iphone10,SamsungFold2
4	HP-ProBook
5	Duke390-Bike
6	Lenovo-ThinkPad
7	i20-Hyundai

```
4 select string_agg(name,',') from products  
5  
6  
7
```

Data Output Explain Messages Notifications Scratch Pad

	string_agg
▲	text
1	chocolate,chips,iphone11,iphone10,iphone9,iphone8,iphone9s,iphone7s,iphone7,LG-Laptop,Samsung-Laptop,MacBook,MacBookAir,HP-P

```
2  
3 select now()
```

Data Output Explain Messages Notifications Scratch Pad

	now	lock
	timestamp with time zone	
1	2021-11-06 15:32:49.349124+05:30	

```
3 select CURRENT_DATE;
```

	Data Output	Explain	Messages	Notifications
	current_date date			
1	2021-11-06			

```
3 select CURRENT_TIME;
```

Data Output Explain Messages Notifications Scratch Pad

	current_time time with time zone	lock
1	15:35:59.209438+05:30	

```
1 -- TO_DATE (text,dateformat)
2 select to_date('2017-01-01','YYYYMMDD')
```

Data Output Explain Messages Notifications Scratch Pad

	to_date date	lock
1	2017-01-01	

```
1 -- TO_DATE (text,dateformat)
2 select to_date('20170101','YYYYMMDD')
```

Data Output Explain Messages Notifications

to_date	date
1	2017-01-01

Query Editor Query History

```
1 -- TO_DATE (text,dateformat)
2 select to_date('01/01/2021','DD/MM/YYYY');
```

Data Output Explain Messages Notifications Scratch Pad

to_date	date
1	2021-01-01

```
1 -- TO_DATE (text,dateformat)
2 select to_date('06 Nov 2021','DD Mon YYYY')
```

Data Output Explain Messages Notifications Scratch Pad

to_date	date
1	2021-11-06

Query Editor Query History

```
1 select distinct extract(hour from updated_at) from products
```

Data Output Explain Messages Notifications Scratch Pad

	date_part	
▲	double precision	🔒
1	0	

Query Editor Query History

```
1 create temp table product_temp as select * from products
```

Data Output Explain Messages Notifications Scratch Pad

SELECT 33

Query returned successfully in 101 msec.

```
6  
7 select length(name),name from products  
8  
9  
10
```

Data Output

Explain

Messages

Notifications

Scratch Pad

	length integer	name character varying	
1	9	chocolate	
2	5	chips	
3	8	iphone11	
4	8	iphone10	
5	7	iphone9	
6	7	iphone8	
7	8	iphone9s	
8	8	iphone7s	

Query Editor Query History

```
4 select * from products where id in (101,103,102,10,105,106,104)
5
6
7
8
```

Data Output Explain Messages Notifications Scratch Pad

	id integer	name character varying	price integer	description text	title character varying (10)
1	102	iphone11	60000	[null]	[null]
2	103	iphone10	50000	[null]	[null]
3	104	[null]	40000	[null]	[null]
4	105	[null]	38000	[null]	[null]
5	106	[null]	55000	[null]	[null]

```
o  
9 select coalesce(name,'NO NAME') from products where id in (101,103,102,10,105,106,104)  
10  
11  
12
```

Data Output Explain Messages Notifications Scratch Pad

	coalesce	lock
	character varying	
1	iphone11	
2	iphone10	
3	NO NAME	
4	NO NAME	
5	NO NAME	

```
9 select coalesce(name,'NO NAME'), coalesce(price,0) from products  
10  
11  
12
```

Data Output Explain Messages Notifications Scratch Pad

	coalesce character varying	lock	coalesce integer	lock
1	iphone11		60000	
2	iphone10		50000	
3	NO NAME		0	
4	NO NAME		0	
5	NO NAME		0	

Window Functions

INTERMEDIATE SQL

