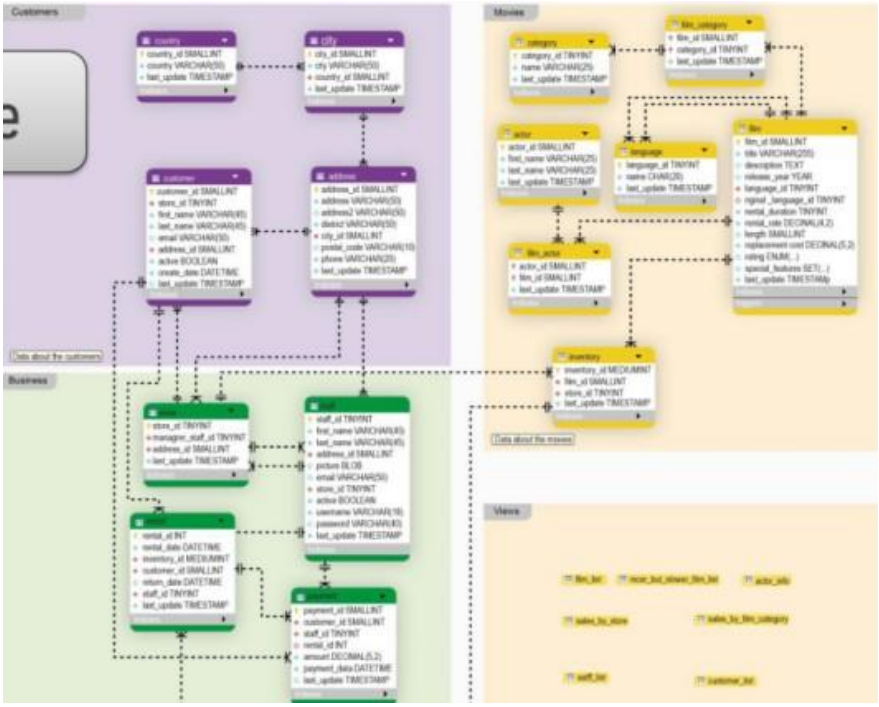
SQL

Day 1-3



Sample Database:

SELECT :

Select * from table;
select column1 from table;
select column1,column2 from table;

ORDER BY :

select * from table
order by column1;

select * from table
order by column1 desc;

select column1, column2 from table
order by column1 desc, column2;

--First data is arranged with column1 in desc if two rows have same column1 then its arranged incolumn2 asc order

```
select first_name,last_name
from customer
order by first_name desc, last_name desc
limit  6
```

Data Output Messages Notifications

	first_name text	last_name text
1	ZACHARY	HITE
2	YVONNE	WATKINS
3	YOLANDA	WEAVER
4	WILMA	RICHARDS
5	WILLIE	MARKHAM
6	WILLIE	HOWELL

SELECT DISTINCT :

select distinct column1
from table;

select distinct column1,column2
from table;
/* when we give two columns it checks if combination is repeated or not
(i.e if (a,b) is firstname and last name.if there is another a,b it wont be displayed
if there is a,c it will be displayed) */

```
SELECT DISTINCT
first_name,
last_name
FROM actor
ORDER BY first_name
```

Data Output Explain Messages Not

	first_name text	last_name text
1	ADAM	GRANT
2	ADAM	HOPPER
3	AL	GARLAND
4	ALAN	DREYFUSS

LIMIT :

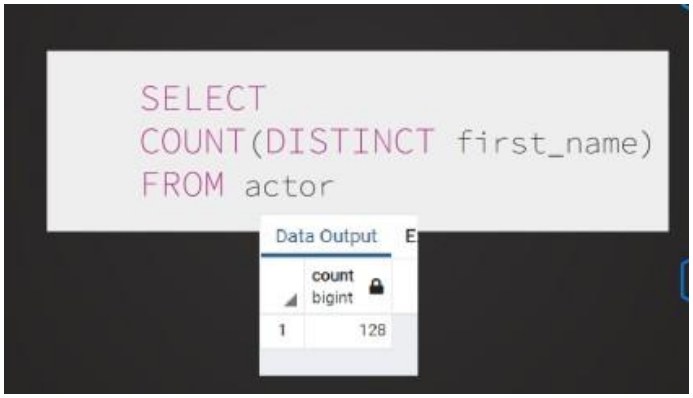
select * from table
limit n;

/* Where n is no of rows
it is always written at end of query */

COUNT :

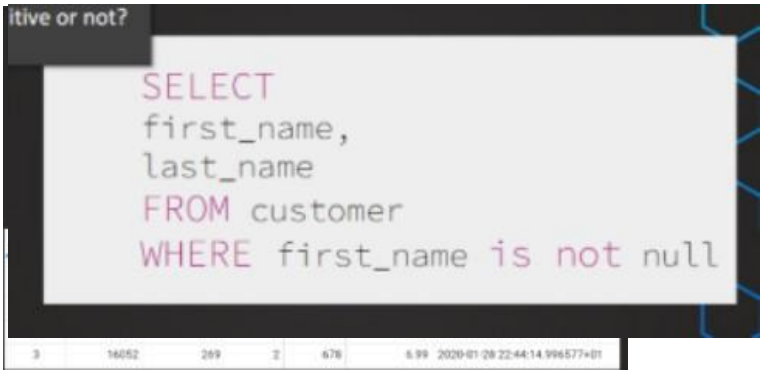
select count(*) from table;
select count(column1) from table;

select count(distinct column1) from table;



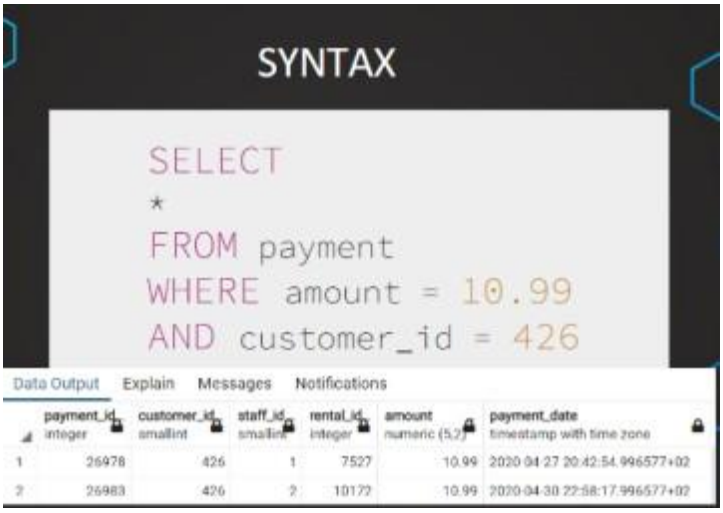
DAY-2 :
WHERE :

-- filter the data , always after from
select * from table column1=some_x;



AND OR :

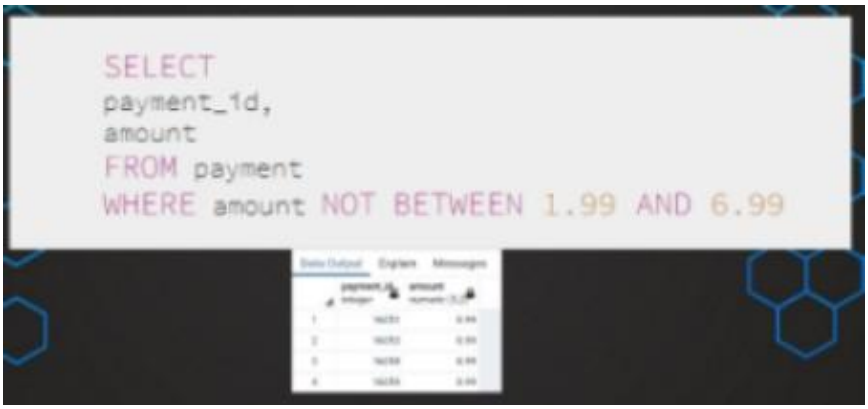
-- Connects conditions



-- normally when both and , or are used in query first and will be executed then or so weusebraces to get or execute first

BETWEEN AND :

--Used to filter a range of values



in above as we wrote 2020-01-26 doesn't give data in that day .we should enter 2020-01-26 23:59 to get data of 26 also.

LIKE :

✓ Used to filter by matching against a pattern
✓ it's case sensitive
✓ Use wildcards: _ any single character
✓ Use wildcards: % any sequence of characters
✓ ILIKE is case insensitive
"_A%" gives a name having second letter A
"A%" gives a string having A as first letter
"%A" gives a string having A as last letter
"%A_" gives a name having last second letter as A

Commenting & Aliases :
✓ Comment to make code more readable & understandable
✓ Use -- Single line comment
✓ Use /*[.....
.....]*/ Multiple lines comment

DAY-3:

AGGRIGATION FUNCTIONS :

Most common aggregation functions
SUM()
AVG()
MIN()
MAX()
COUNT()

ROUND():
round(column,2)-rounds off values in column by 2 digits

We can use multiple aggregate functions together :
SELECT SUM(amount),
COUNT(*),AVG(amount)
FROM payment
We can not use one aggregate function and a column to diplay like this:
SELECT
SUM(amount) ,
payment_id
FROM payment

Output:

```
ERROR:  column "payment.payment_id" must appear in the GROUP BY clause
or be used in an aggregate function
LINE 3: payment_id
        ^

SQL state: 42803
Character: 21Query returned successfully in 130 msec.
```

but we can diplay by using group by

```
SELECT
SUM(amount)
payment_id
FROM payment
group by payment_id;
```

Output:

Data Output Messages Notifications

	sum numeric	payment_id integer
1	3.99	22344
2	5.99	30052
3	4.99	26264
4	2.99	24416

Date :

-it converts the date with timestamp to date

Challenges:
In 2020, April 28, 29 and 30 were days with very high revenue.
That's why we want to focus in this task only on these days
(filter accordingly).
Find out what is the average payment amount grouped by
customer and day – consider only the days/customers with
more than 1 payment (per customer and day).
Order by the average amount in a descending order.

Ans:

```
select customer_id
,date(payment_date) dday,round(avg(amount),2) avg_amount,count(*)
from payment
where  date(payment_date) in ('2020-04-28','20202-04-29','2020-04-30')
group by dday,customer_id
having  count(*)>1
order  by avg_amount desc;
```

Day 4-6

Day 4 :

Upper :
converts strings to upper case

```
select upper(email)
from customer
```

Lower:
```
select lower(email)
from customer
```

Left:

```
 left(string,n)

 select left(email,2) from customer;
--gives first 2 letters of email
```

Right:

```
 right(string,n)

 select right(email,2) from customer;
--gives last 2 letters of email

select right(left(email,2),1) from customer;
-- gives 2 nd letter
```

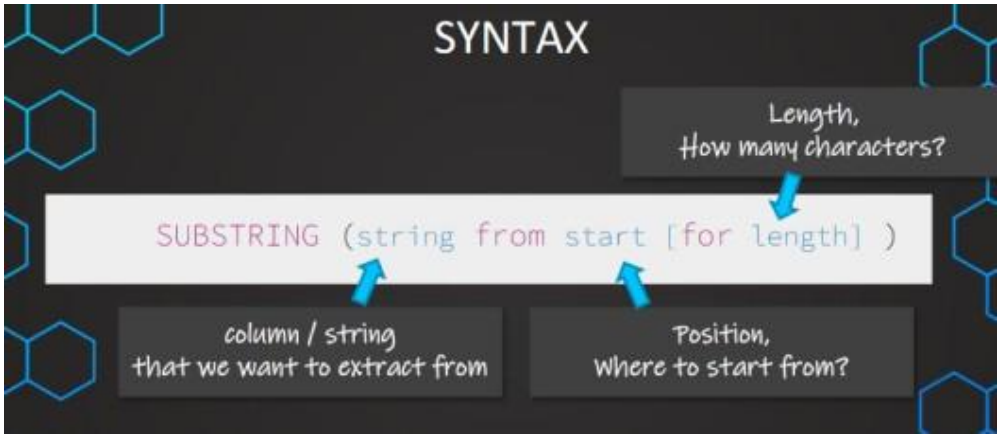
Concatinate: || is used

```
select left(first_name) ||'.'||left(last_name)
from customer;
```

Position:

position('.' in email)
gives position of the given character in given string

Substring:
gives a substring of given string



SUBSTRING (email from POSITION ('.' in email)+1 for 3)
gives 3 character after . in email

You need to create an anonymized form of the email addresses in the following way:

1	M***.S***@sakilacustomer.org
2	P***.J***@sakilacustomer.org

In a second query create an anonymized form of the email addresses in the following way:

EXTRACT

Field	Extract from timestamp/date
CENTURY	century
DAY	day of month (1-31)
DECADE	decade that is year divided by 10
DOW	day of week Sunday (0) to Saturday (6)
DOY	day of year that ranges from 1 to 366
EPOCH	number of seconds since 1970-01-01 00:00:00 UTC
HOUR	hour (0-23)
ISODOW	day of week based on ISO 8601 Monday (1) to Sunday (7)
ISOYEAR	ISO 8601 week number of year
MICROSECONDS	seconds field, including fractional parts, multiplied by 1000000
MILLENNIUM	millennium
MILLISECONDS	seconds field, including fractional parts, multiplied by 1000
MINUTE	minute (0-59)
MONTH	month (1-12)
QUARTER	quarter of year
SECOND	second
TIMEZONE	timezone offset from UTC, measured in seconds
TIMEZONE_HOUR	hour component of time zone offset
TIMEZONE_MINUTE	minute component of time zone offset
WEEK	number of ISO 8601 week-numbering week of year
YEAR	year

Challenge:

You need to analyze the payments and find out the following:

- What's the month with the highest total payment amount?
- What's the day of week with the highest total payment amount? (0 is Sunday)
- What's the highest amount one customer has spent in a week?

Write a SQL query to find out!

Answer:

```
select extract(month from payment_date),
sum(amount)
from payment
group by extract(month from payment_date)
order by sum (amount) desc;
```

```
select extract(dow from payment_date),
sum(amount)
from payment
group by extract (dow from payment_date )
order by sum(amount) desc;

select extract(week from payment_date),
sum(amount),customer_id
from payment
group by customer_id,extract (week from payment_date )
order by sum(amount) desc;
```

To_char:

Used to get custom formats timestamp/date/numbers

TO_CHAR (date/time/interval, format)

Example
TO_CHAR (rental_date, 'MM-YYYY')

Challenge:

You need to create a list for the suppcity team of all rental durations of customer with customer_id 35.

Also you need to find out for the suppcity team which customer has the longest average rental duration?

Answer:

```
select
return_date-rental_date
from rental
where customer_id=35
group by customer_id;

select
avg(return_date-rental_date)
from rental
group by customer_id
order by avg(return_date-rental_date) ;
```

Day 5:

Mathematical functions and operators

Operator	Description	Example	Result
+	addition	4 + 3	7
-	subtraction	5 - 3	2
*	multiplication	4 * 2	8
/	division (integer division truncates the result)	8 / 4	2
%	modulo (remainder)	10 % 4	2
^	exponentiation	2 ^ 3	8

Mathematical functions and operators

Function	Description	Example	Result
abs(x)	absolute value	abs(-7)	7
round(x,d)	round x to d decimal places	round(4.3543)	4.35
ceiling(x)	round up to integer	ceiling(4.3543)	5
floor(x)	round down to integer	floor(4.3543)	4

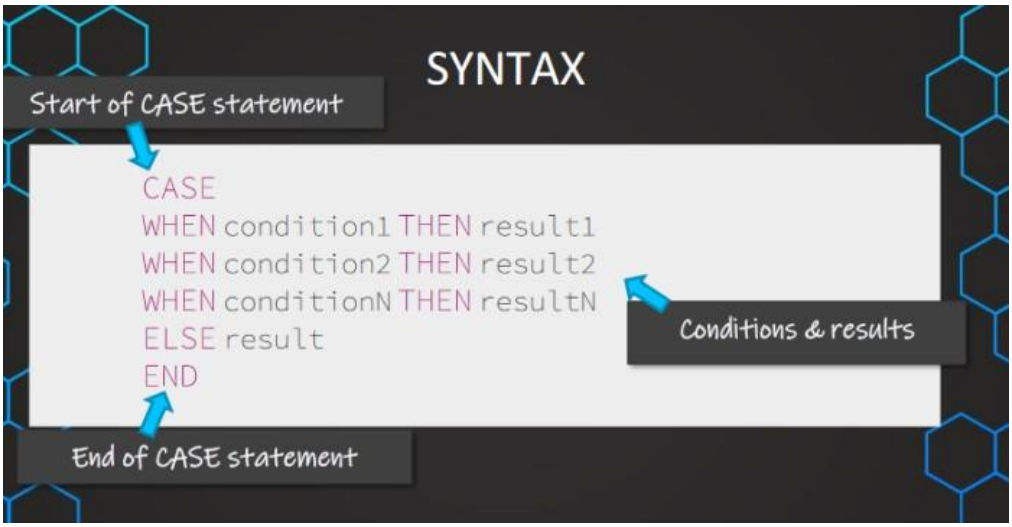
Challenge:
Your manager is thinking about increasing the prices for films that are more expensive to replace.
For that reason, you should create a list of the films including the relation of rental rate / replacement cost where the rental rate is less than 4% of the replacement cost.
Create a list of that film_ids together with the percentage rounded to 2 decimal places. For example 3.54 (=3.54%)

Ans:
```
select film_id,
round(rental_rate/replacement_cost*100,2) as rate
from film
where
round(rental_rate/replacement_cost*100,2)<4
order by rate ;
```

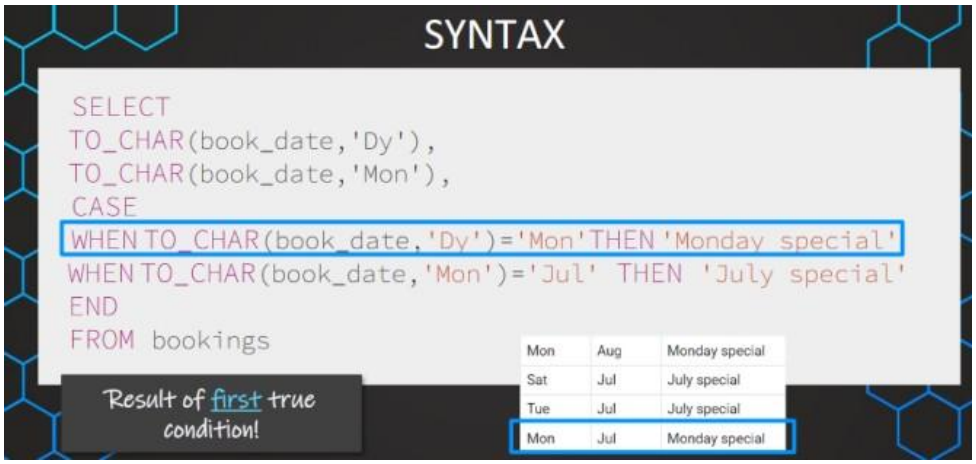
Case When:

Like IF/THEN statement

Goes through a set of conditions
returns a value if a condition is met



--if a row satisfies two conditions then first condition output will be shown.



--if no condition is satisfied then null will be returned
--ELSE => result if no condition is met

Challenge:
You need to find out how many tickets you have sold in the following categories:
• Low price ticket: total_amount < 20,000
• Mid price ticket: total_amount between 20,000 and 150,000
• High price ticket: total_amount >= 150,000
How many high price tickets has the company sold?

```
select count(*),
case
        when total_amount<20000 then 'low price ticket'
        when total_amount<150000 then 'mid price ticket'
        else 'high price ticket'
end  price
from bookings
group by price
```

```
order by count(*) desc;
```

Challenge:
You need to find out how many flights have departed in the following seasons:
- Winter: December, January, February
- Spring: March, April, May
- Summer: June, July, August
- Fall: September, October, November

```
select count(*),
case
when extract(month from scheduled_departure)
            in (12,01,02) then 'Winter'
when extract(month from scheduled_departure)
            in (03,04,05) then 'Spring'
when extract(month from scheduled_departure)
            in (06,07,08) then 'Summer'
else 'Fall'
end season
from flights
group by season;
```

You want to create a tier list in the following way:
1. Rating is 'PG' or 'PG-13' or length is more then 210 min:
'Great rating or long (tier 1)
2. Description contains 'Drama' and length is more than 90min:
'Long drama (tier 2)'
3. Description contains 'Drama' and length is not more than 90min:
'Shcity drama (tier 3)'
4. Rental_rate less than $1:
'Very cheap (tier 4)'
If one movie can be in multiple categories it gets the higher tier assigned.
How can you filter to only those movies that appear in one of these 4 tiers?

```
select title,
case
when rating in ('PG','PG-13') or length>210
then 'Great rating or long(tier 1)'
when description like '%Drama%' and length>90
then 'Long drama(tier 2)'
when description like '%Drama%' then 'Short drama(tier 3)'
when rental_rate<1
then 'Very cheap(tier 4)'
end tier
from film
where case
when rating in ('PG','PG-13') or length>210
then 'Great rating or long(tier 1)'
when description like '%Drama%' and length>90
```

then 'Long drama(tier 2)'
when description like '%Drama%' then 'Short drama(tier 3)'
when rental_rate<1
then 'Very cheap(tier 4)'
end is not null;

Coalesce:

replaces null value in a string with given string
Syntax:
COALESCE (actual_arrival, '1970-01-01 0:00')

CAST:

Changes the data type of a value
Syntax
CAST (value/column AS data type)

Replace:
REPLACE
✓ Replaces text from a string in a column with another text
Syntax:
REPLACE (column, old_text, new_text)

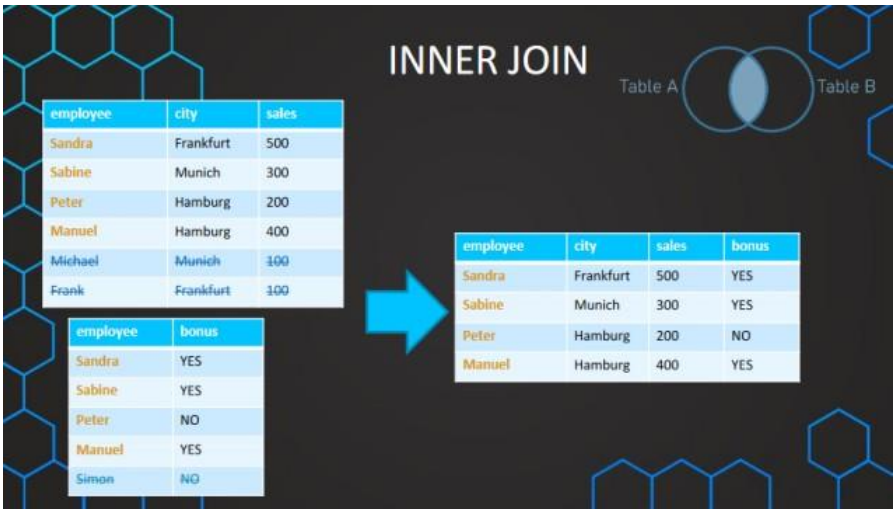
Challenge:
Change flight numbers into integer

```
select
cast(replace (flight_no,'PG','') as int)
from flight;
```

Day 6:

JOINS:
✓ Combine information from multiple tables in one query

INNER JOIN:



Challenge:

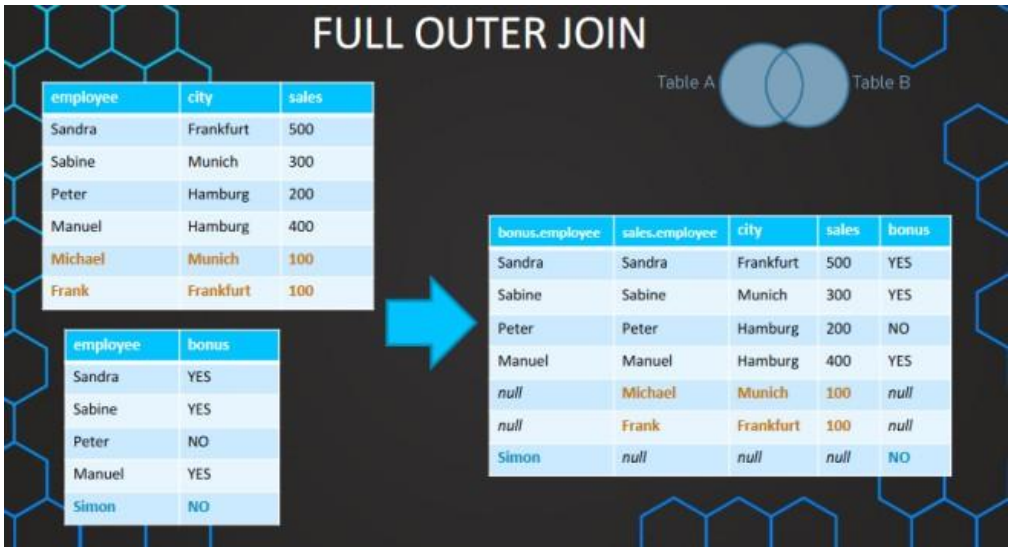
The airline company wants to understand in which category they sell most tickets.

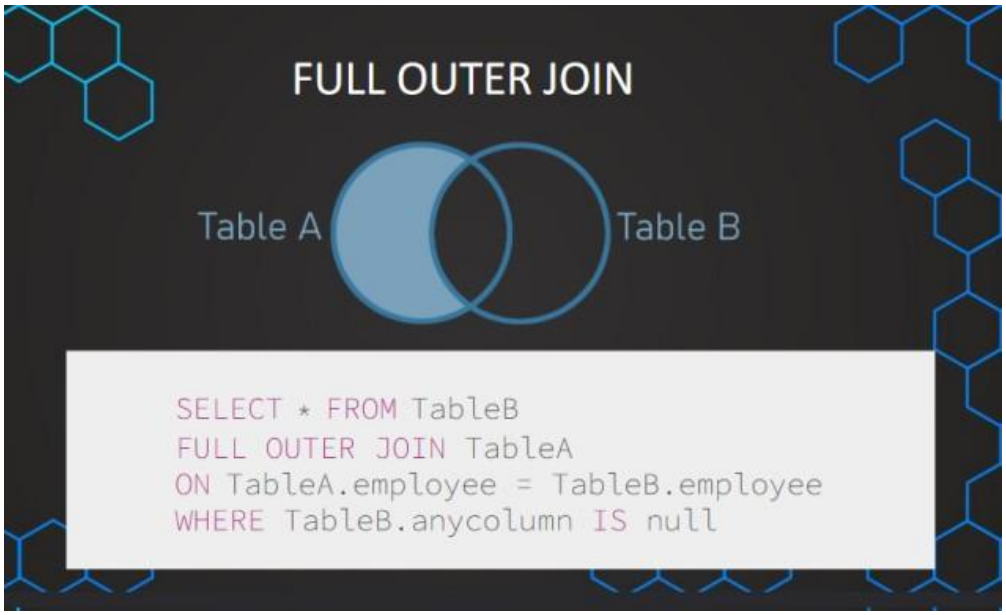
How many people choose seats in the category
- Business
- Economy or
- Comfort?

You need to work on the seats table and the boarding_passes table.

```
select fare_conditions ,count(*)
from boarding_passes a
inner join seats b on a.seat_no=b.seat_no
group by fare_conditions
order by count(*) desc;
```

Full outer Join:

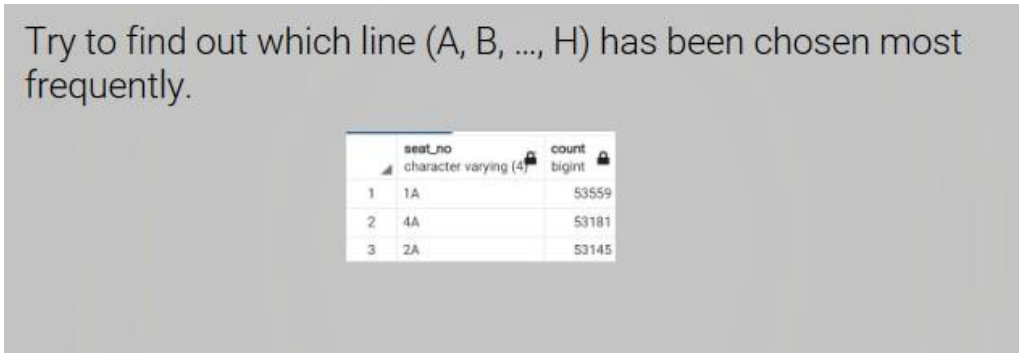




Left Outer Join:

Challenge:
The flight company is trying to find out what their most popular seats are.
Try to find out which seat has been chosen most frequently.
Make sure all seats are included even if they have never been booked.
Are there seats that have never been booked?

```
select b.seat_no, count(*) from
boarding_passes a left  join
seats b on a.seat_no=b.seat_no
group by b.seat_no
order by count(*) desc;
```




```
select right(b.seat_no,1) line ,count(*)
from boarding_passes a left join
seats b on a.seat_no=b.seat_no
group by line
order by count(*) desc;
```

The company wants to run a phone call campaing on all customers in Texas (=district).
What are the customers (first_name, last_name, phone number and their

district) from Texas?

```
select first_name,last_name,phone,
district
from customer a join address b
on a.address_id=b.address_id
where district='Texas';
```

Are there any (old) addresses that are not related to any customer?

```
select * from customer a right join
address b on a.address_id = b.address_id
where a.address_id is null;
```

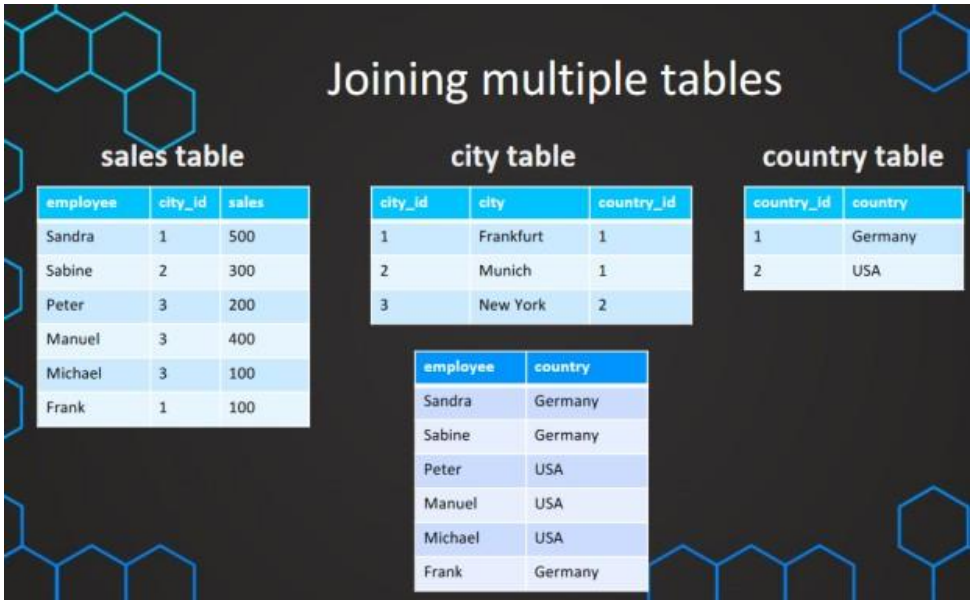
Multiple join conditions:

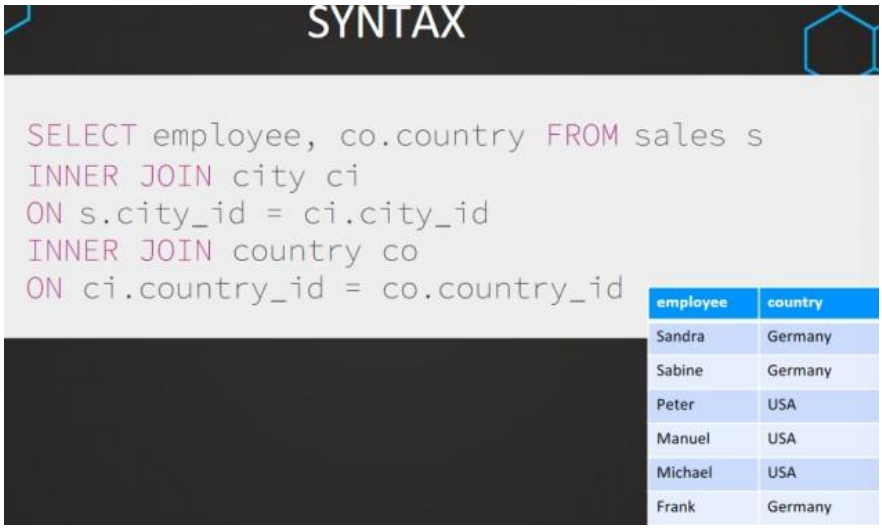
```
SELECT * FROM TableA a
INNER JOIN TableB b
ON a.first_name = b.first_name
AND a.last_name = b.last_name
```

```
SELECT * FROM TableA a
INNER JOIN TableB b
ON a.first_name = b.first_name
AND a.last_name = 'Jones'
```

```
select seat_no,round(avg(amount),2)
from boarding_passes b left join
ticket_flights t on
b.ticket_no= t.ticket_no
and b.flight_id=t.flight_id
group by seat_no
order by 2 desc;
```

Joining multiple tables:





Challenge:

The company wants customize their campaigns to customers depending on the country they are from.

Which customers are from Brazil?
Write a query to get first_name, last_name, email and the country from all customers from Brazil.

```
select first_name,last_name,email,country
from customer cu inner join address ad
on cu.address_id=ad.address_id
inner join city ci on ci.city_id=ad.city_id
inner join country co on co.country_id=ci.country_id
where country='Brazil';
```

Which title has GEORGE LINTON rented the most often?

Answer:

```
select title ,count(*) from film f
inner join inventory i on f.film_id=i.film_id
inner join rental  r on  i.inventory_id=r.inventory_id
inner join customer c on r.customer_id= c.customer_id
where first_name='GEORGE'and last_name='LINTON'
group by title
order by count(*) desc
limit 1;
```

CADDYSHACK JEDI - 3 times.

Which passenger (passenger_name) has spent most amount in their bookings (total_amount)?
Answer:

```
select passenger_name ,sum(total_amount)
from tickets t inner join
bookings b on t.book_ref=b.book_ref
group by passenger_name
order by sum(total_amount) desc
```

limit 1;

ALEKSANDR IVANOV with 80964000.

Which fare_condition has ALEKSANDR IVANOV used the most?
Answer:

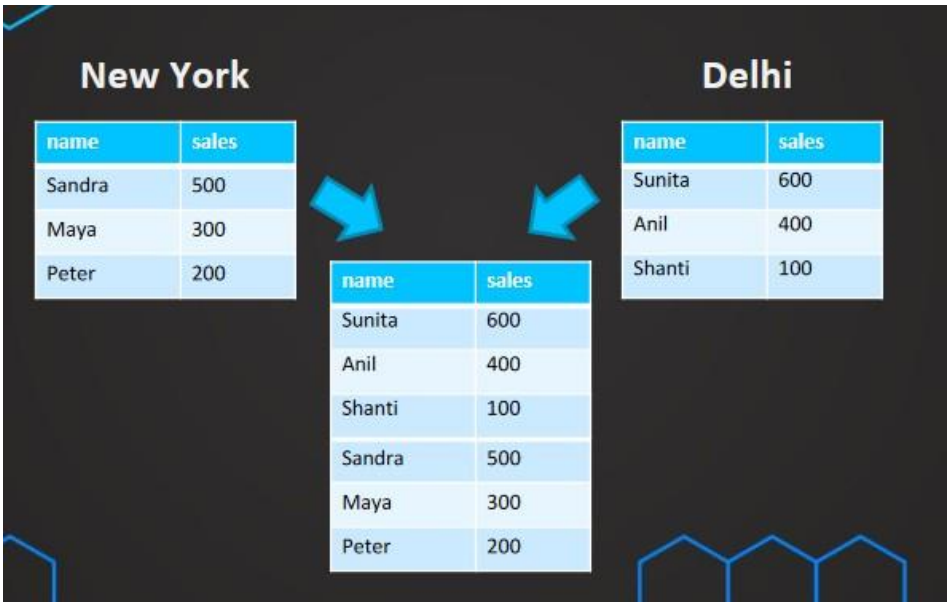
```
select fare_conditions, count(*)
from ticket_flights tf inner join
tickets t on tf.ticket_no=t.ticket_no
where passenger_name='ALEKSANDR IVANOV'
group by fare_conditions
order by count(*) desc
limit 1;
```

Economy 2131 times.

Day 7-9

Day 7:

Union:



SYNTAX :

```
SELECT first_name, sales FROM vancouver
UNION
SELECT first_name, sales FROM delhi

--duplicates are not displayed
--need to use union all for displaying them
--The order matches the column!
--Data types must match!
```

Sub Queries:

• sub queries must return single column only

-In where statment
```
select * from film
where length>(select avg(length) from film);
```
-In from statment
```
select round(avg(total_id),2) from
(select sum(amount) total_id,date(payment_date)
from payment group by date(payment_date)) sub;
```

-In select statment
```
select *,
(select max(amount)from payment)-amount as diff
from payment ;
```

Challenge:

1. Return the details of the films whose length is more than the average length of films.
Ans:
```
select * from film
where length>(select avg(length) from film);
```

2. Return th details of the films available in inventory 2 more than 3 times.
Ans:
```
select * from film
where film_id in (select film_id
from inventory
where store_id=2
group by film_id
having  count(*)>3);
```

3. Return all customer's first_names and last names that have made a payment on '2020-01-25'.
Ans:
```
select first_name,last_name
from customer
where customer_id in (select customer_id
                  from payment where
                   date(payment_date)
                      ='2020-01-25');
```

4. Return all customer's first_names and email that have spent more than $30.
Ans:
```
select first_name,email
from customer
where customer_id in (select customer_id
                  from payment
                group by customer_id
                     having  sum(amount)>30);
```

5. Return all customer's first_names and last names that are from California and have spent more than 100 in total.

Ans:

```
select first_name,email
from customer
where customer_id in (select customer_id
                  from payment
                group by customer_id
                    having sum(amount)>100)
and customer_id in (select customer_id
                  from address a inner join
                  customer c on a.address_id=c.address_id
                   where district='California');
```

6. What is the average spent per day (average daily revenue)?

Ans:

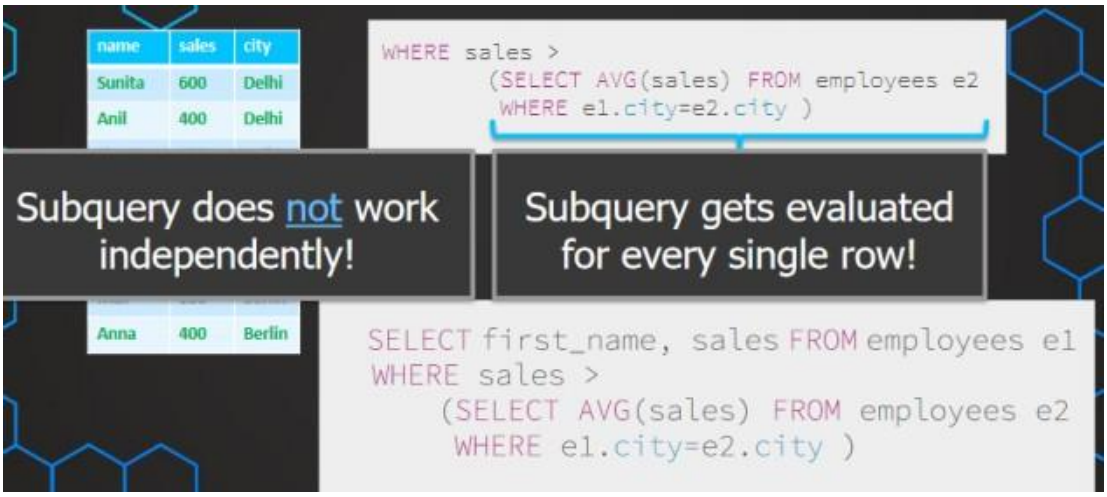
```
select round(avg(total_amount),2) from
(select sum(amount) total_amount,date(payment_date)
from payment group by date(payment_date)) sub;
```

7. Show all the payments together with how much the payment amount is below the maximum payment amount.

Ans :

```
select *,
(select max(amount)from payment)-amount as diff
from payment ;
```

Correlated Subqueries:



Challenge:

1. Show only those movie titles, their associated film_id and replacement_cost with the lowest replacement_costs for in each rating category – also show the rating.

Ans:
```
select title,film_id,replacement_cost,rating
from film f1 where
replacement_cost=(select min(replacement_cost)
                from film f2
                where f1.rating=f2.rating);
```

2. Show only those movie titles, their associated film_id and the length that have the highest length in each rating category – also show the rating.

Ans:
```
select title,film_id,length,rating
from film f1 where
length=(select max(length)
                from film f2
                where f1.rating=f2.rating);
```

3. Show the payment details with highest amount spent by each customer.

Ans:
```
select * from payment p1
where amount=(select max(amount)
            from payment p2
            where p1.customer_id=p2.customer_id)
order by customer_id;
```

Day-8:

go to "solutions.sql" for practise questions.

Day-9:

Data Defination:
Create
Alter
Drop

Data Manipulation:
Insert
Update
Delete

Creating database:
CREATE DATABASE <database_name>;

Dropping database:
DROP DATABASE <database_name>;

Data Types:

Numeric:

Type	Storage size	Range	Notes
INT	4 bytes	-2147483648 to +2147483647	Typical choice
SMALLINT	2 bytes	-32768 to +32767	Small integers
BIGINT	8 bytes	-9223372036854775808 to +9223372036854775807	Large integers
DECIMAL	variable	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point	user-defined precision
SERIAL	variable	1 to 2147483647	autoincrementing integer

numeric is same as Decimal

numeric(precision, scale)
Precision: total count of digits
Scale: count of decimal places
24.99 2 decimal places
numeric(4,2)

Strings:

Type	Storage size	Example	Notes
character varying(n), varchar(n)	variable-length with limit	Any text, "Hello"	Less flexible to change!
character(n), char(n)	fixed-length, blank padded	"M" or "F"	Not better in performance!
text	variable unlimited length	Any text, "Hello"	Winner!

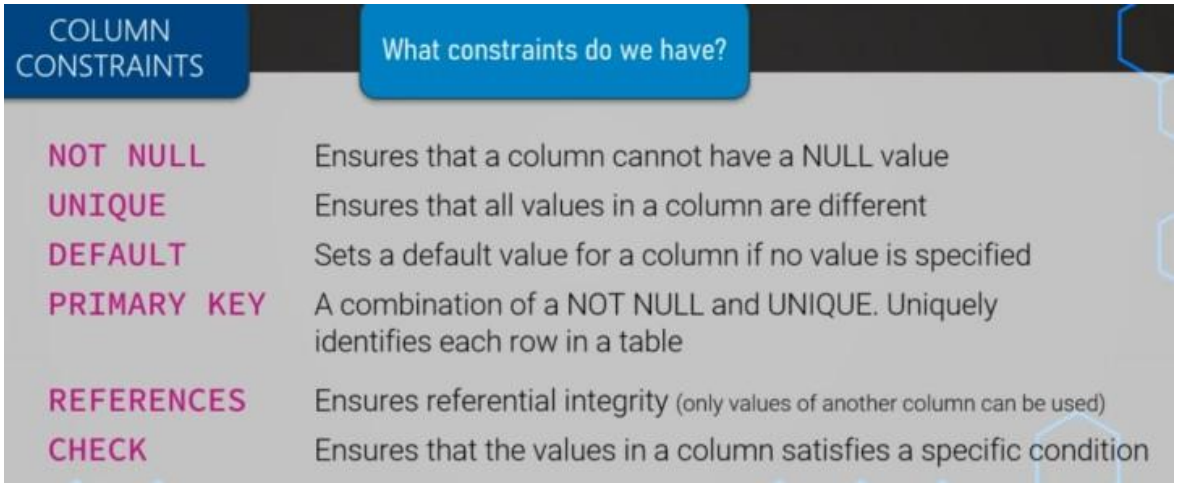
Date/Time:

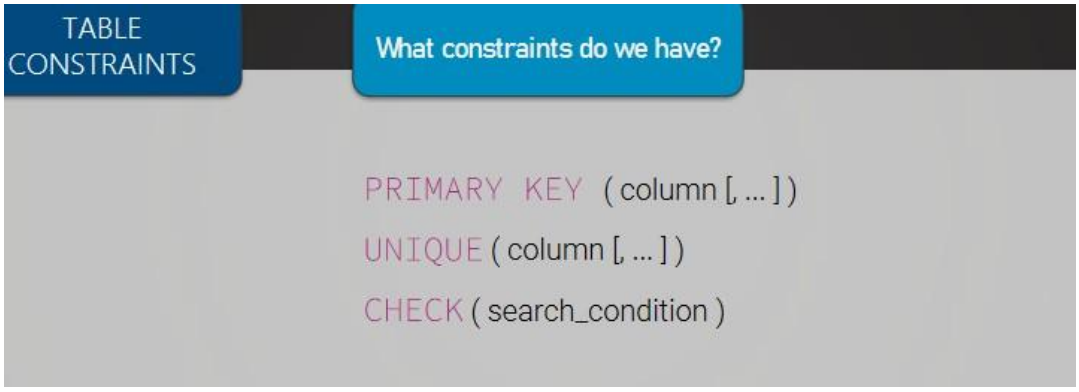
Type	Description	Example
date	Just date without time	'2022-11-28'
time (with/without time zone)	Just time without date	'01:02:03.678'
timestamp (with/without time zone)	Date and time	'2022-11-28 01:02:03.678+02'
intervals	Time interval	'3 days 01:02:03.678'

Others:

Type	Description	Example	Range
Boolean	state of true or false	is_in_stock	TRUE,FALSE, NULL
Enum	A value of a list of ordered values	movie_rating	User-defined
array	Stores a list of value	text[] or int[]	Depending on type

Constraints:





Create Table:

```
CREATE TABLE staff(
staff_id SERIAL PRIMARY KEY,
name VARCHAR(50) NOT NULL
UNIQUE(name,staff_id)
)
```

Drop Table:

```
DROP TABLE IF EXISTS directors
```

Insert:

INSERT

```
INSERT INTO online_sales
(customer_id,amount)
VALUES (269,10.99)
```

```
INSERT INTO online_sales
VALUES  (1,269,13,10.99,'BUNDLE2022')
```

Alter Table:

1. ADD, DELETE columns
2. ADD, DROP constraints
3. RENAME columns
4. ALTER data types

Drop:

```
ALTER TABLE <table_name>
DROP COLUMN <column_name>
```

Truncate Table:

```
TRUNCATE TABLE <table_name>
```

Add:

```
ALTER TABLE staff
ADD COLUMN IF NOT EXISTS date_of_birth DATE
```

Change Data-type:

```
ALTER TABLE <table_name>
ALTER COLUMN <column_name> TYPE NEW_TYPE
```

Rename:

```
ALTER TABLE old_table_name
RENAME TO new_table_name

ALTER TABLE <table_name>
RENAME COLUMN <old_column_name> TO <new_column_name>
```

Default value:

```
ALTER TABLE <table_name>
ALTER COLUMN <column_name> SET DEFAULT <value>

ALTER TABLE <table_name>
ALTER COLUMN <column_name> DROP NOT NULL
```

Add constraint:

```
ALTER TABLE <table_name>
ADD CONSTRAINT <constraint_name>,
ADD PRIMARY KEY(column1,column2[,…])
```

Multiple changes:

```
ALTER TABLE director
ALTER COLUMN director_account_name SET DEFAULT 3,
ALTER COLUMN first_name TYPE TEXT,
ALTER COLUMN last_name TYPE TEXT,
ADD COLUMN middle_name TEXT,
ADD CONSTRAINT constraint_1 UNIQUE(account_name)
```

Check:

Limit the value range that can be placed in a column

CHECK

```
CREATE TABLE <table_name> (
<column_name>  TYPE  CHECK(condition))
```

```
CREATE TABLE director
(name TEXT,
date_of_birth DATE,
start_date DATE,
end_date DATE CHECK(start_date > '01-01-2000'))
```

Adding check constraint by using alter:

```
ALTER TABLE director
ADD CONSTRAINT date_check CHECK(start_date < end_date )
```

Drop check constraint:

```
ALTER TABLE director
DROP CONSTRAINT date_check
```

Challenge:

To create director table:

```
create table director (director_id serial primary key,
        director_account_name varchar(20)  unique,
       first_name  varchar(50),last_name varchar(50) default
        'not specified' ,date_of_birth date,
       address_id  int references address(address_id)
                    );
```

ALTER DIRECTOR TABLE steps

1. director_account_name to VARCHAR(30)

```
alter table director
alter director_account_name type varchar(30);
```

2. drop the default on last_name

```
alter table director
alter last_name drop default;
```

3. add the constraint not null to last name

```
alter table director
alter last_name set not null;
```

4. add the column email of data type VARCHAR (40)

```
alter table director
add email varchar(40);
```

5. rename the director_account_name to account_name

```
alter table director
rename director_account_name to account_name;
```

6. rename the table from director to directors

```
alter table director
rename to Directors;
```

Challenge:

Create a table called *songs* with the following columns:

song_id	song_name	genre	price	release_date
[PK] integer	character varying (30)	character varying (30)	numeric (4,2)	date

1. During creation add the DEFAULT 'Not defined' to the genre.
2. Add the not null constraint to the *song_name* column
3. Add the constraint with default name to ensure the price is at least 1.99.
4. Add the constraint *date_check* to ensure the release date is between today and 01-01-1950.
5. Try to insert a row like this:

	song_id	song_name	genre	price	release_date
	[PK] integer	character varying (30)	character varying (30)	numeric (4,2)	date
1	4	SQL song	Not defined	0.99	2022-01-07

6. Modify the constraint to be able to have 0.99 allowed as the lowest possible price.
7. Try again to insert the row.

```
create table song(song_id int primary key ,song_name
                varchar(30),genre varchar(30),
               price   numeric(4,2),release_date date);
```

1. alter table song
   alter genre set default 'Not defined'

2. alter table song
   alter song_name set not null

3. alter table song
   add constraint c1 check(price>=1.99)

4. alter table song
   add constraint date_check check(date(release_date)
   between '01-01-1950'and current_date)

5. alter table song
   drop constraint c1

6. alter table song
   add constraint c1 check(price>=0.99)

7. insert into song (song_id,song_name,genre,price,release_date)
   values (4,'SQL song','not defined',0.99,'2023-01-07')

Day-8

```
--Create a list of all the different (distinct) replacement costs of the films.

select distinct replacement_cost from film
order by replacement_cost;

/* Write a query that gives an overview of how many films have replacements costs
in the following cost ranges
   low: 9.99 - 19.99

   medium: 20.00 - 24.99

   high: 25.00 - 29.99 */

select count(*),
case
when replacement_cost between 9.99 and 19.99
then 'low_cost'
when replacement_cost between 19.99 and 24.99 then 'meddium'
when replacement_cost between 24.99 and 29.99 then 'high'
end cost
from film
group by cost;

/*Create a list of the film titles including their title,
 length and category name ordered descendingly by the length.
  Filter the results to only the movies
in the category 'Drama' or 'Sports'.*/
```

```
select title,length,name
from film f inner join film_category fc
on f.film_id=fc.film_id
inner join category c on fc.category_id=c.category_id
where name in('Drama','Sports')
order by length desc;

/*Create an overview of how many movies (titles)
there are in each category (name). */

select name,count(*)
from film f inner join film_category fc
on f.film_id=fc.film_id
inner join category c on fc.category_id=c.category_id
group by name
order by 2 desc;

/*Create an overview of the actors first and last names and
in how many movies they appear.*/
select first_name,last_name,count(*)
from film_actor fa inner join actor a
on fa.actor_id= a.actor_id
group by first_name,last_name
order by count(*) desc;

/*Create an overview of the addresses that are not
associated to any customer.*/

select * from customer c right join address a
on c.address_id=a.address_id
where customer_id is null;

/*Create an overview of the cities and how much sales
(sum of amount) have occured there.*/

select sum(amount),city from payment p join customer c on
p.customer_id=c.customer_id join address a
on a.address_id=c.address_id join city ci on
a.city_id=ci.city_id
group by city
order by 1 desc;

/* Create an overview of the revenue (sum of amount) grouped by
a column in the format "country, city".*/

select   sum(amount),country||','||city
from payment p join customer c on
p.customer_id=c.customer_id join address a
on a.address_id=c.address_id join city ci on
a.city_id=ci.city_id join country co
```

```
on   ci.country_id=co.country_id
group by city,country
order by 1 ;

/*Create a list with the average of the sales amount
each staff_id has per customer.*/

select  staff_id,round(avg(sum),2)
from (select staff_id,sum(amount) as sum
from payment
 group by customer_id,staff_id) sub
 group by staff_id;

/*Create a query that shows average daily
revenue of all Sundays.*/

select avg(sum)
from (select sum(amount) sum from payment
        where extract(dow from payment_date)=0
     group by date(payment_date),
      extract(dow from payment_date)) sub;

/*Create a list of movies - with their length and their
replacement cost - that are longer than the average length
in each replacement cost group.*/

select title,length,f1.replacement_cost
from film f1
where length>(select avg(length)
            from film f2
            where f1.replacement_cost=f2.replacement_cost)
order by length;

/*Create a list that shows how much the average customer
spent in total (customer life-time value) grouped by the
different districts.*/

select round(avg(sum),2),district from
(select customer_id,sum(amount) sum
                            from payment
                            group by customer_id) b
join customer c on c.customer_id=b.customer_id
join address a on a.address_id=c.address_id
group by district
order by 1 desc;

/* Create a list that shows all payments including the payment_id,
amount and the film category (name) plus the total amount
that was made in this category. Order the results ascendingly
by the category (name) and as second order criterion by the
```

```
payment_id ascendingly*/

select amount,name,payment_id,
(select sum(amount)from payment p
join rental r on r.rental_id=p.rental_id
join inventory i on i.inventory_id=r.inventory_id
join film_category fc on fc.film_id=i.film_id
join category c on c.category_id=fc.category_id
where c1.name=c.name)
from payment p
join rental r on r.rental_id=p.rental_id
join inventory i on i.inventory_id=r.inventory_id
join film_category fc on fc.film_id=i.film_id
join category c1 on c1.category_id=fc.category_id
order by name,payment_id;

/* Create a list with the top overall revenue of
a film title (sum of amount per title) for each
category (name).*/

select title,name,sum(amount) from payment p
join rental r on r.rental_id=p.rental_id
join inventory i on i.inventory_id=r.inventory_id
join film f on f.film_id=i.film_id
join film_category fc on fc.film_id=i.film_id
join category c1 on c1.category_id=fc.category_id
group by name,title
having
sum(amount)=(select max(am)from
            ( select sum(amount) am from payment p
join rental r on r.rental_id=p.rental_id
join inventory i on i.inventory_id=r.inventory_id
join film f on f.film_id=i.film_id
join film_category fc on fc.film_id=i.film_id
join category c on c.category_id=fc.category_id
where c.name=c1.name
group by name,title )sub)
```

Day 10-12

Day 10:

Udate:
UPDATE <table>
SET <column>=value

--udate specific rows
UPDATE songs
SET genre='Pop music'
WHERE song_id=4

Challenge:

Update all rental prices that are 0.99 to 1.99.
The customer table needs to be altered as well:
1. Add the column initials (data type varchar(10))
2. Update the values to the actual initials for example Frank Smith should be F.S

Update film
set rental_rate=1.99
where rental_rate=0.99

alter table customer
add column initials varchar(10)

update customer
set initials=left(first_name,1)||'.'||left(last_name,1)

Delete :

DELETE FROM payment
WHERE payment_id in (17064,17067)
 RETURNING *

Create table as:

CREATE TABLE customer_anonymous
AS
SELECT customer_id, initials
FROM customer
WHERE first_name LIKE 'C%'

--Pyhsical storage needed!
--Data change wont be reflected in the table

CREATE VIEW … AS:

```
CREATE VIEW customer_anonymous
AS
SELECT customer_id, initials
FROM customer
WHERE first_name LIKE 'C%'

--no Physical storage
--Data change wil be reflected
```

Materialized view:

```
CREATE VIEW customer_anonymous
AS
SELECT customer_id, initials
FROM customer
WHERE first_name LIKE 'C%'

--when you want update the chages in table in view
REFRESH MATERIALIZED VIEW <view_name>
```

Managing views:

```
ALTER VIEW:
        ALTER VIEW v_customer_info
        RENAME COLUMN name TO customer_name
DROP VIEW:
        DROP VIEW customer_anonymous
ALTER MATERIALIZED VIEW:
        ALTER MATIRIALIZED VIEW v_customer_info
        RENAME COLUMN name TO customer_name
DROP MATERIALIZED VIEW:
        DROP MATERIALIZED VIEW customer_anonymous
CREATE OR REPLACE VIEW:
        CREATE OR REPLACE VIEW v_customer_info
        AS new_query
```

Challenge:
Display name of customer and his total spendings

```
Create table as total_spendings
select first_name||' '||last_name as name,sum(amount) as total_amount
from customer c left join payment p
on c.customer_id=p.customer_id
group by name
```

Create a view called *films_category* that shows a list of the film titles including their title, length and category name ordered descendingly by the length.
Filter the results to only the movies in the category 'Action' and 'Comedy'

```
create view as films_category
```

```
select title,length,name
from film f left join film_category fc
on f.film_id=fc.film_id left join
category c on fc.category_id=c.category_id
where name in ('Action','Comedy')
order by length desc
```

In this challenge we use again the view *v_customer_info* lik this first:

```
CREATE VIEW v_customer_info
AS
SELECT cu.customer_id,
   cu.first_name || ' ' || cu.last_name AS name,a.address,
   a.postal_code,
   a.phone,
   city.city,
   country.country
    FROM customer cu
    JOIN address a ON cu.address_id = a.address_id
    JOIN city ON a.city_id = city.city_id
    JOIN country ON city.country_id = country.country_id
ORDER BY customer_id
```

You need to perform the following tasks:

1) Rename the view to v_customer_information.
2) Rename the customer_id column to c_id.
3) Add also the initial column as the third column to the view by replacing the view.

```
alter view v_customer_info
rename to v_customer_information

alter view v_customer_information
rename customer_id to c_id

drop view v_customer_information

CREATE OR REPLACE VIEW v_customer_information
AS
SELECT cu.customer_id,
   cu.first_name || ' ' || cu.last_name AS name,cu.initials,
       a.address,
   a.postal_code,
   a.phone,
   city.city,
   country.country
    FROM customer cu
    JOIN address a ON cu.address_id = a.address_id
    JOIN city ON a.city_id = city.city_id
    JOIN country ON city.country_id = country.country_id
ORDER BY customer_id
```

Import & Export:

Import external data into an
existing table

Table needs to be created first!
Data needs to be in correct format!

Export data from a table
into a csv file

Day 11:
Window functions:
these doesn't effect no. of rows
Example:

Over()& partition by:
it acts like group by but doesnt change row

Challenge:

Write a query that returns the list of movies including
- film_id,
- title,
- length,
- category,
- average length of movies in that category.

Order the results by film_id.

Result

film_id integer	title text	category text	length_of_movie smallint	avg_length_in_category numeric
1	ACADEMY DINOSAUR	Documentary	86	108.75
2	ACE GOLDFINGER	Horror	48	112.48
3	ADAPTATION HOLES	Documentary	50	108.75
4	AFFAIR PREJUDICE	Horror	117	112.48
5	AFRICAN EGG	Family	130	114.78

```
select f.film_id,title,length ,name as category,
avg(length) over(partition by name) from film
f  left join  film_category fc on  f.film_id=fc.film_id
left join category c on fc.category_id=c.category_id
order by 1;
```

Write a query that returns all payment details including
- the number of payments that were made by this customer and that amount

Order the results by payment_id.

Result

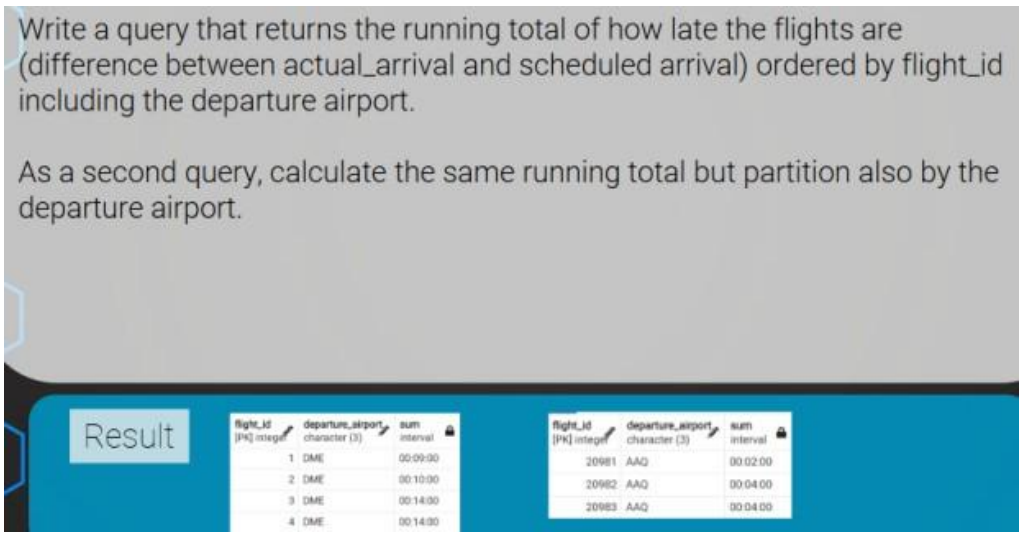
payment_id integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)	payment_date timestamp with time	no_payments_with_that_amount bigint
16050	269	2	7	1.99	2020-01-24 22:40:...	1
16051	269	1	98	0.99	2020-01-25 16:16:...	3
16052	269	2	678	6.99	2020-01-28 22:44:...	5

```
select *,
count(*) over(partition by amount,customer_id)
from payment
order by customer_id;
```

over() & order by:

it adds or counts by ordering the given column values

Challenge:

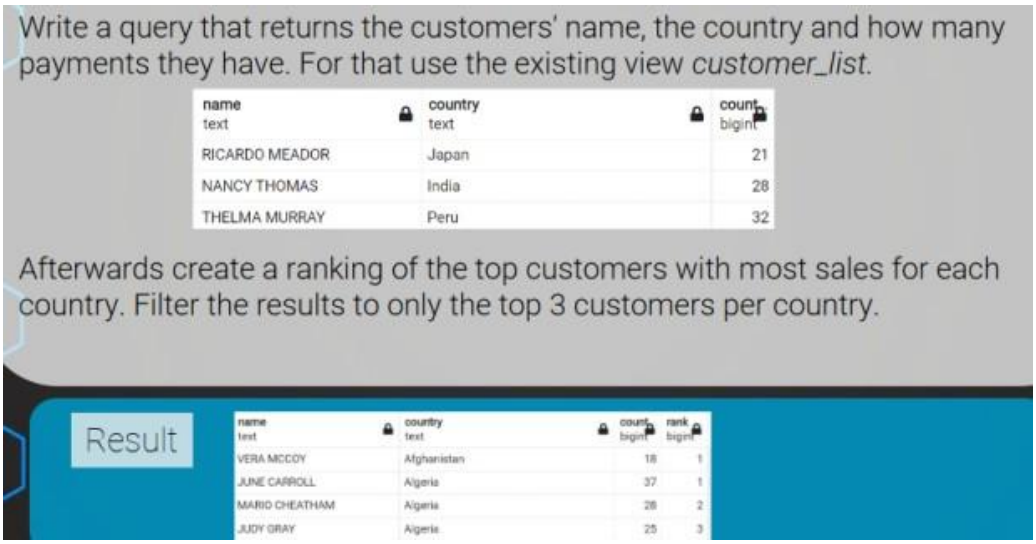



```
select flight_id,departure_airport,
sum(actual_arrival-scheduled_arrival)
over(order by flight_id) from flights;

select flight_id,departure_airport,
sum(actual_arrival-scheduled_arrival)
over(partition by departure_airport
    order by flight_id) from flights;
```

Rank():
It gives ranking to the given column
There are two types :
- DENSE_RANK(): continous ranking
- RANK(): uncontinous ranking

Challenge



```
select name,country,count(*)
from customer_list cl left join payment p on
cl.id=p.customer_id
group by name,country;

select * from (select name,country,count(*),
dense_rank() over(partition by country
            order by count(*)  desc)  rank
from customer_list cl left join payment p on
cl.id=p.customer_id
group by name,country) a
where rank between 1 and 3;
```

FIRST_VALUE():

```
gives first value of the group select name,country,count(*),
first_value(count(*)) over(partition by country
            order by count(*) ) rank
from customer_list cl left join payment p on
cl.id=p.customer_id
group by name,country;
```

name text	country text	count bigint	rank bigint
VERA MCCOY	Afghanistan	18	18
JUDY GRAY	Algeria	25	25
MARIO CHEATHAM	Algeria	28	25
JUNE CARROLL	Algeria	37	25
ANTHONY SCHWAB	American Samoa	20	20
CLAUDE HERZOG	Angola	25	25
MARTIN BALES	Angola	27	25
BOBBY BOUDREAU	Anguilla	35	35
DARRYL ASHCRAFT	Argentina	23	23
PERRY SWAFFORD	Argentina	24	23

Here Angola's first value(ie lowest value) is written for all rows with angola as its country

LEAD&LAG:

lead gives succeding value

```
select name,country,count(*),
lead(count(*)) over(partition by country
            order by count(*) desc) rank,
count(*)-lead(count(*)) over(partition by country
            order by count(*) desc) diff
from customer_list cl left join payment p on
cl.id=p.customer_id
group by name,country;
```

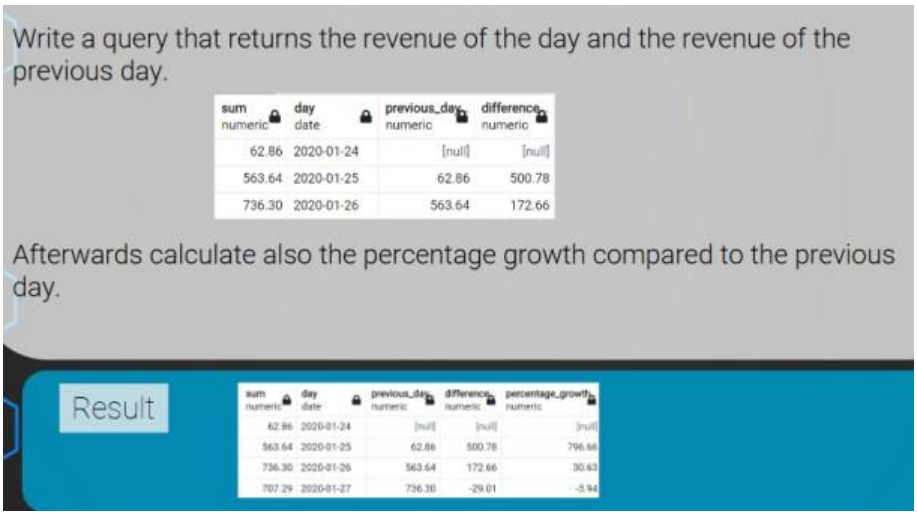
name text	country text	count bigint	rank bigint	diff bigint
VERA MCCOY	Afghanistan	18	[null]	[null]
JUNE CARROLL	Algeria	37	28	9
MARIO CHEATHAM	Algeria	28	25	3
JUDY GRAY	Algeria	25	[null]	[null]
ANTHONY SCHWAB	American Samoa	20	[null]	[null]
MARTIN BALES	Angola	27	25	2
CLAUDE HERZOG	Angola	25	[null]	[null]
BOBBY BOUDREAU	Anguilla	35	[null]	[null]
LEONARD SCHOFIELD	Argentina	32	32	0
JULIA FLORES	Argentina	32	30	2

lag gives preceding values

```
select name,country,count(*),
lag(count(*)) over(partition by country
            order by count(*) desc) rank, count(*)-
lag(count(*)) over(partition by country
            order by count(*) desc) diff
from customer_list cl left join payment p on
cl.id=p.customer_id
group by name,country;
```

name text	country text	count bigint	rank bigint	diff bigint
VERA MCCOY	Afghanistan	18	[null]	[null]
JUNE CARROLL	Algeria	37	[null]	[null]
MARIO CHEATHAM	Algeria	28	37	-9
JUDY GRAY	Algeria	25	28	-3
ANTHONY SCHWAB	American Samoa	20	[null]	[null]
MARTIN BALES	Angola	27	[null]	[null]
CLAUDE HERZOG	Angola	25	27	-2
BOBBY BOUDREAU	Anguilla	35	[null]	[null]
LEONARD SCHOFIELD	Argentina	32	[null]	[null]
JULIA FLORES	Argentina	32	32	0

Challenge:



```
select date(payment_date) as day ,sum(amount),
lag(sum(amount)) over(order by date(payment_date)) previous_day,
sum(amount)-lag(sum(amount))
over(order by date(payment_date)) difference
from payment
group by date(payment_date);

select date(payment_date) as day ,sum(amount),
lag(sum(amount)) over(order by date(payment_date)) previous_day,
sum(amount)-lag(sum(amount))
over(order by date(payment_date)) difference,
round((sum(amount)-lag(sum(amount))
over(order by date(payment_date)))/lag(sum(amount))
      over(order by date(payment_date))*100,2) percentage
from payment
group by date(payment_date);
```

Advanced Challenges:

Write a query that calculates now the share of revenue each staff_id makes per customer. The result should look like this:

first_name text	last_name text	staff_id smallint	total numeric	percentage numeric
AARON	SELBY	[null]	110.76	100.00
AARON	SELBY	1	63.86	57.66
AARON	SELBY	2	46.90	42.34
ADAM	GOOCH	[null]	101.78	100.00
ADAM	GOOCH	1	51.89	50.98
ADAM	GOOCH	2	49.89	49.02

```
select first_name,last_name,staff_id,sum(amount),
round(sum(amount)/first_value(sum(amount)) over(partition by
        first_name,last_name order by sum(amount) desc)*100,2) percentage
from customer c left join payment p
on c.customer_id=p.customer_id
group by grouping sets((first_name,last_name),((first_name,last_name),staff_id))
```

DAY-12:

Grouping Sets:

```
 select to_char(payment_date,'Month') as month,staff_id,
sum(amount)
from payment
group by grouping
sets( (staff_id),(month),(staff_id,mo
nth)) order by month, staff_id;
```

month text	staff_id smallint	sum numeric
April	1	14241.43
April	2	14497.40
April	[null]	28738.83
February	1	5027.29
February	2	5095.45
February	[null]	10122.74
January	1	2700.91
January	2	2276.68
January	[null]	4977.59

The above query finds sum by grouping only staff_id,grouping only month and then by grouping both month,staff_id

Challenge:


```
select first_name,last_name,
staff_id,sum(amount)
from payment p right join customer c
 on p.customer_id=c.customer_id
 group by grouping
 sets( (first_name,last_name),(staff_i
 d),
    ((first_name,last_name),staff_id))
 order by first_name,last_name,staff_id;
```

Rollup:

```
select 'Q'||to_char(payment_date,'Q') as quarter,
extract(month from payment_date) as month,
date (payment_date),sum(amount)
from payment
group by
rollup(
'Q'||to_char(payment_date,'Q'),
extract(month from payment_date),
date (payment_date))
order by 1,2,3;
```

13	Q1	2	2020-02-18	1566.52
14	Q1	2	2020-02-19	1535.50
15	Q1	2	2020-02-20	1482.70
16	Q1	2	2020-02-21	1302.01
17	Q1	2	2020-02-22	121.72
18	Q1	2	[null]	10122.74
19	Q1	3	2020-03-01	2258.75
20	Q1	3	2020-03-02	3033.22
21	Q1	3	2020-03-03	301.32
22	Q1	3	2020-03-17	2193.91
23	Q1	3	2020-03-18	2896.60
24	Q1	3	2020-03-19	2680.88
25	Q1	3	2020-03-20	2876.60
26	Q1	3	2020-03-21	2879.54
27	Q1	3	2020-03-22	2700.78
28	Q1	3	2020-03-23	2724.73
29	Q1	3	2020-03-24	361.23
30	Q1	3	[null]	24907.56
31	Q1	[null]	[null]	40007.89
32	Q2	4	2020-04-06	2013.39

Roll up gives heirarcy from down to top
by seeing above query we can see that first we get sum of amount in each day in a month and quater then total sum of that month then sum of days in next month and so on then sum of the quarter then in next quarter sum of amount in each day in a month then total sum of month and so on end with sum of amount in next quartor.

Challenge:

Write a query that calculates a booking amount rollup for the hierarchy of quarter, month, week in month and day.

quarter numeric	month numeric	week_in_month text	day date	booking_amount numeric
2	6	3	2017-06-21	441900.00
2	6	3	[null]	441900.00
2	6	4	2017-06-22	775300.00
2	6	4	2017-06-23	1822000.00

```
 select to_char(book_date,'Q') as quarter,
extract(month from book_date) as month,
to_char(book_date,'w') week,
 date (book_date),sum(total_amount)
 from bookings
 group by
 rollup(
 to_char(book_date,'Q') ,
extract(month from book_date) ,
to_char(book_date,'w'),
 date (book_date))
```

order by 1,2,3,4 ;

Cube:

```
GROUP BY
CUBE  (column1, column2, column3)
```

```
GROUP BY
GROUPING SETS  (
(column1, column2, column3),
(column1, column2),
(column1, column3),
(column2, column3),
(column1),
(column2),
(column3),
()
)
```

It gives all possible combinations groupings
We use it when there is no natural hireracy

```
select customer_id,staff_id,
date(payment_date),sum(amount)
from payment
group by
cube( customer_id,staff_id,
date(payment_date))
order by 1,2,3;
```

1	2	2020-04-27	2.99
1	2	2020-04-28	6.98
1	2	2020-04-29	2.99
1	2	2020-04-30	2.99
1	2	[null]	57.85
1	[null]	2020-01-25	2.99
1	[null]	2020-01-28	1.99
1	[null]	2020-02-15	7.98
1	[null]	2020-02-16	14.98
1	[null]	2020-02-18	6.98
1	[null]	2020-02-21	3.99
1	[null]	2020-03-01	4.99

Challenge:
Write a query that returns all grouping sets in all combinations of customer_id, date and title with the aggregation of the payment amount.
The desired result looks like this:

How do you order the output to get that desired result?

```
select  p.customer_id,date(payment_date),
title,sum(amount) from
payment p left join rental r
on p.rental_id=r.rental_id
left join inventory i
on  i.inventory_id=r.inventory_id
left join film f on f.film_id=i.film_id
group by
 cube(p.customer_id,date(payment_date),
title)
order by 1,2,3
```

SELF JOIN:

```
select emp.employee_id,emp.name as employee,mng.name
as maneger,mng2.name as manager2 from employee emp left join
employee mng on emp.manager_id=mng.employee_id
left join employee mng2 on mng.manager_id=mng2.employee_id
```

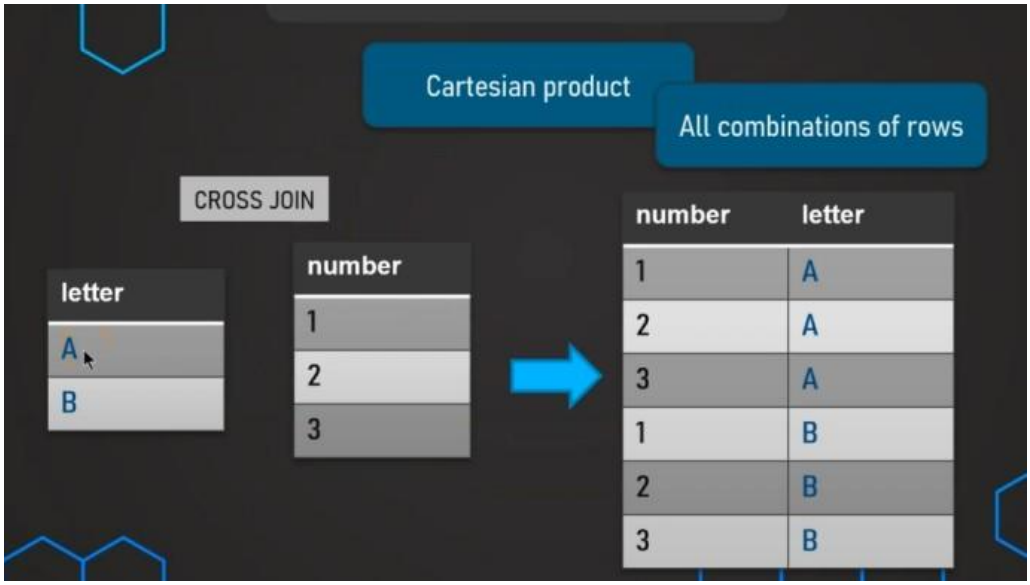
customer_id smallint	date date	title text	total numeric
1	2020-01-25	PATIENT SISTER	2.99
1	2020-01-25	[null]	2.99
1	2020-01-28	TALENTED HOMICIDE	0.99
1	2020-01-28	[null]	0.99
1	2020-02-15	DETECTIVE VISION	0.99
1	2020-02-15	FERRIS MOTHER	9.99

Challenge:

	employee_id integer	employee character varying (50)	maneger character varying (50)	manager2 character varying (50)
1	1	Liam Smith	[null]	[null]
2	2	Oliver Brown	Liam Smith	[null]
3	3	Elijah Jones	Liam Smith	[null]
4	4	William Miller	Liam Smith	[null]
5	5	James Davis	Oliver Brown	Liam Smith
6	6	Olivia Hernandez	Oliver Brown	Liam Smith
7	7	Emma Lopez	Oliver Brown	Liam Smith
8	8	Sophia Andersen	Oliver Brown	Liam Smith
9	9	Mia Lee	Elijah Jones	Liam Smith
10	10	Ava Robinson	Elijah Jones	Liam Smith


```
select  f1.title  ,f2.title,f1.length
from film f1  join film f2
on   f1.length=f2.length
where  f1.title  !=f2.title
order by 3 desc
```

Cross Join:




```
select staff_id,store.store_id,first_name from
staff cross join store
```

staff_id integer	store_id integer	first_name text
1	1	Mike
1	2	Mike
2	1	Jon
2	2	Jon

Natural Join:

Just like a normal JOIN

Automatically joins using columns
with the same column name

```
SELECT
*
FROM payment
NATURAL JOIN customer
```

This doesn't give proper result if we have more than one column with matching name

Example:

```
select * from customer
natural inner join address
```

in this we wont get any result as customer ,address has two
common columns address_id,last_update

Day-13

1. In your company there hasn't been a database table with all the employee information yet.
You need to set up the table called employees in the following way:

emp_id	first_name	last_name	job_position	salary	start_date	birth_date	store_id	department_id	manager_id
[PK] integer	text	text	text	numeric (8,2)	date	date	integer	integer	integer

There should be NOT NULL constraints for the following columns:
first_name,
last_name ,
job_position,
start_date DATE,
birth_date DATE

```
create table employees(
emp_id int primary key,first_name varchar(20) not null
,last_name varchar(20) not null ,job_position varchar(20)
not null,salary decimal(8,2), start_date date not null,
birth_date date not null,store_id int,department_id int,
manager_id int)
```

Set up an additional table called departments in the following way:

department_id	department	division
[PK] integer	text	text

Additionally no column should allow nulls.

create table departments (
department_id int primary key ,department varchar(20)
not null,division varchar(20) not null)

2. Alter the employees table in the following way:
- Set the column department_id to not null.
- Add a default value of CURRENT_DATE to the column start_date.
- Add the column end_date with an appropriate data type (one that you think makes sense).
- Add a constraint called birth_check that doesn't allow birth dates that are in the future.
- Rename the column job_position to position_title.

alter table employees
alter department_id set not null,
alter start_date set default current_date,
add column end_date date,
add constraint birth_check
Check((current_date-birth_date)>0)

alter table employees
rename column job_position to position_title

3. Insert the following values into the *employees* table.
There will be most likely an error when you try to insert the values.
So, try to insert the values and then fix the error.
Columns:
(emp_id,first_name,last_name,position_title,salary,start_date,birth_date,store_id,department_id,manager_id,end_date)

--At first problem is end date in given values is not in date format so add '' to them
Insert into employees (emp_id,first_name,last_name,position_title,salary,start_date,birth_date,store_id,department_id,manager_id,end_date)
values
(1,'Morrie','Conaboy','CTO',21268.94,'2005-04-30','1983-07-10',1,1,NULL,NULL),
(2,'Miller','McQuarter','Head of BI',14614.00,'2019-07-23','1978-11-09',1,1,1,NULL),
(3,'Christalle','McKenny','Head of Sales',12587.00,'1999-02-05','1973-01-09',2,3,1,NULL),
(4,'Sumner','Seares','SQL Analyst',9515.00,'2006-05-31','1976-08-03',2,1,6,NULL),
(5,'Romain','Hacard','BI Consultant',7107.00,'2012-09-24','1984-07-14',1,1,6,NULL),
(6,'Ely','Luscombe','Team Lead Analytics',12564.00,'2002-06-12','1974-08-01',1,1,2,NULL),
(7,'Clywd','Filyashin','Senior SQL Analyst',10510.00,'2010-04-05','1989-07-23',2,1,2,NULL),
(8,'Christopher','Blague','SQL Analyst',9428.00,'2007-09-30','1990-12-07',2,2,6,NULL),
(9,'Roddie','Izen','Software Engineer',4937.00,'2019-03-22','2008-08-30',1,4,6,NULL),
(10,'Ammamaria','Izhak','Customer Support',2355.00,'2005-03-17','1974-07-27',2,5,3,'2013-04-14'),

(11,'Carlyn','Stripp','Customer Support',3060.00,'2013-09-06','1981-09-05',1,5,3,NULL),
(12,'Reuben','McRorie','Software Engineer',7119.00,'1995-12-31','1958-08-15',1,5,6,NULL),
(13,'Gates','Raison','Marketing Specialist',3910.00,'2013-07-18','1986-06-24',1,3,3,NULL),
(14,'Jordanna','Raitt','Marketing Specialist',5844.00,'2011-10-23','1993-03-16',2,3,3,NULL),
(15,'Guendolen','Motton','BI Consultant',8330.00,'2011-01-10','1980-10-22',2,3,6,NULL),
(16,'Doria','Turbat','Senior SQL Analyst',9278.00,'2010-08-15','1983-01-11',1,1,6,NULL),
(17,'Cort','Bewlie','Project Manager',5463.00,'2013-05-26','1986-10-05',1,5,3,NULL),
(18,'Margarita','Eaden','SQL Analyst',5977.00,'2014-09-24','1978-10-08',2,1,6,'2020-03-16'),
(19,'Hetty','Kingaby','SQL Analyst',7541.00,'2009-08-17','1999-04-25',1,2,6,NULL),
(20,'Lief','Robardley','SQL Analyst',8981.00,'2002-10-23','1971-01-25',2,3,6,'2016-07-01'),
(21,'Zaneta','Carlozzi','Working Student',1525.00,'2006-08-29','1995-04-16',1,3,6,'2012-02-19'),
(22,'Giana','Matz','Working Student',1036.00,'2016-03-18','1987-09-25',1,3,6,NULL),
(23,'Hamil','Evershed','Web Developper',3088.00,'2022-02-03','2012-03-30',1,4,2,NULL),
(24,'Lowe','Diamant','Web Developper',6418.00,'2018-12-31','2002-09-07',1,4,2,NULL),
(25,'Jack','Franklin','SQL Analyst',6771.00,'2013-05-18','2005-10-04',1,2,2,NULL),
(26,'Jessica','Brown','SQL Analyst',8566.00,'2003-10-23','1965-01-29',1,1,2,NULL)

Insert the following values into the departments table.

department_id [PK] integer	department text	division text
1	Analytics	IT
2	Finance	Administration
3	Sales	Sales
4	Website	IT
5	Back Office	Administration

```
INSERT INTO departments
VALUES (1, 'Analytics','IT'),
(2, 'Finance','Administration'),
(3, 'Sales','Sales'),
(4, 'Website','IT'),
(5, 'Back Office','Administration')
```

4. Jack Franklin gets promoted to 'Senior SQL Analyst' and the salary raises to 7200. Update the values accordingly.

```
UPDATE employees
SET position_title = 'Senior SQL Analyst'
WHERE emp_id=25;

UPDATE employees
SET salary=7200
WHERE emp_id=25;
```

The responsible people decided to rename the position_title Customer Support to Customer Specialist. Update the values accordingly.

```
UPDATE employees
SET position_title='Customer Specialist'
WHERE position_title='Customer Support';
```

All SQL Analysts including Senior SQL Analysts get a raise of 6%.
Upate the salaries accordingly.

```
UPDATE employees
SET salary=salary*1.06
WHERE position_title LIKE '%SQL Analyst';
```

What is the average salary of a SQL Analyst in the company (excluding Senior SQL Analyst)?

```
SELECT ROUND(AVG(salary),2) FROM employees
WHERE position_title='SQL Analyst'
```

5. Write a query that adds a column called manager that contains first_name and last_name (in one column) in the data output.
Secondly, add a column called is_active with 'false' if the employee has left the company already, otherwise the value is 'true'.

```
SELECT
emp.*,
CASE WHEN emp.end_date IS NULL THEN 'true'
ELSE 'false'
END as is_active,
mng.first_name ||' '|| mng.last_name AS manager_name
FROM employees emp
LEFT JOIN employees mng
ON emp.manager_id=mng.emp_id;
```

Create a view called v_employees_info from that previous query.

```
create view v_employees_info
as
SELECT
emp.*,
CASE WHEN emp.end_date IS NULL THEN 'true'
ELSE 'false'
END as is_active,
mng.first_name ||' '|| mng.last_name AS manager_name
FROM employees emp
LEFT JOIN employees mng
ON emp.manager_id=mng.emp_id;
```

6. Write a query that returns the average salaries for each positions with appropriate roundings.

```
select round(avg(salary),2),position_title
from employees
group by position_title
```

7.Write a query that returns the average salaries per division.

```
select round(avg(salary),2),division
from employees natural left join departments
group by division
```

8. Write a query that returns the following:
emp_id,
first_name,
last_name,
position_title,
salary
and a column that returns the average salary for every job_position.
Order the results by the emp_id.

emp_id [PK] integer	first_name text	last_name text	position_title text	salary numeric (8,2)	avg_position_sal numeric
1	Morrie	Conaboy	CTO	21268.94	21268.94
2	Miller	McQuarter	Head of BI	14614.00	14614.00
3	Christalle	McKenny	Head of Sales	12587.00	12587.00
4	Sumner	Seares	SQL Analyst	10085.90	8834.75

```
select emp_id,first_name,last_name,
position_title,salary,
round(avg(salary) over(partition by
   position_title ),2) as avg_position_salary
from employees
order by emp_id
```

How many people earn less than there avg_position_salary?
Write a query that answers that question.
Ideally the output just shows that number directly.

```
select count(*)
from employees e1,(select emp_id,
round(avg(salary) over(partition by
   position_title ),2) as avg_position_salary
from employees) e2
Where e1.emp_id=e2.emp_id and
salary<avg_position_salary
```

9. Write a query that returns a running total of the salary development ordered by the start_date. In your calculation, disregard that fact that people have left the company (write the query as if theywere still working for the company).

emp_id [PK] integer	salary numeric (8,2)	start_date date	avg_pos_sal numeric
12	7119.00	1995-12-31	7119.00
3	12587.00	1999-02-05	19706.00
6	12564.00	2002-06-12	32270.00
20	9519.86	2002-10-23	41789.86

s

```
elect emp_id,salary,start_date,
sum(salary) over(order by start_date)
      avg_pos_salary
from employees
order by 4
```

10. Create the same running total but now also considder the fact that people were leaving the company.

```
SELECT start_date,
SUM(salary) OVER(ORDER BY start_date)
FROM (SELECT
emp_id,salary,start_date
FROM employees
UNION
SELECT emp_id,-salary,end_date
FROM v_employees_info
WHERE is_active ='false'
ORDER BY start_date) a
```

11. Write a query that outputs only the top earner per position_title including first_name and position_title and their salary.

first_name text	position_title text	salary numeric (8,2)
Morrie	CTO	21268.94
Miller	Head of BI	14614.00
Christalle	Head of Sales	12587.00
Ely	Team Lead Analytics	12564.00

```
SELECT first_name,
position_title,salary
FROM employees e1
WHERE salary = (SELECT MAX(salary)
                FROM employees e2
                WHERE e1.position_title=e2.position_title)
```

Add also the average salary per position_title.

first_name text	position_title text	salary numeric (8,2)	avg_in_pos numeric
Morrie	CTO	21268.94	21268.94
Miller	Head of BI	14614.00	14614.00
Christalle	Head of Sales	12587.00	12587.00

```
SELECT
first_name,
position_title,
salary,
(SELECT ROUND(AVG(salary),2) as avg_in_pos FROM employees e3
WHERE e1.position_title=e3.position_title)
FROM employees e1
WHERE salary = (SELECT MAX(salary)
                FROM employees e2
                WHERE  e1.position_title=e2.position_title)
```

Remove those employees from the output of the previous query that have the same salary as the average of their position_title.
These are the people that are the only ones with their position_title.

```
SELECT
first_name,
position_title,
salary,
(SELECT ROUND(AVG(salary),2) as avg_in_pos FROM employees e3
WHERE e1.position_title=e3.position_title)
FROM employees e1
WHERE salary = (SELECT MAX(salary)
                FROM employees e2
                WHERE  e1.position_title=e2.position_title)
AND salary<>(SELECT ROUND(AVG(salary),2) as avg_in_pos FROM employees e3
WHERE e1.position_title=e3.position_title)
```

12. Write a query that returns all meaningful aggregations of
- sum of salary,
- number of employees,
- average salary

grouped by all meaningful combinations of
- division,
- department,
- position_title.

Consider the levels of hierarchies in a meaningful way.

division text	department text	position_title text	sum numeric	count bigint	round numeric
Administration	Back Office	Customer Specialist	5415.00	2	2707.50
Administration	Back Office	Project Manager	5463.00	1	5463.00
Administration	Back Office	Software Engineer	7119.00	1	7119.00
Administration	Back Office	[null]	17997.00	4	4499.25

```
SELECT
division,
department,
position_title,
SUM(salary),
COUNT(*),
ROUND(AVG(salary),2)
FROM employees
NATURAL JOIN departments
GROUP BY
ROLLUP(
division,
department,
position_title
)
ORDER BY 1,2,3
```

13. Write a query that returns all employees (emp_id) including their position_title, department their salary and the rank of that salary partitioned by department.
The highest salary per division should have rank 1.

emp_id integer	position_title text	department text	salary numeric (8,2)	rank bigint
1	CTO	Analytics	21268.94	1
2	Head of BI	Analytics	14614.00	2
6	Team Lead Analytics	Analytics	12564.00	3
7	Senior SQL Analyst	Analytics	11140.60	4

```
SELECT emp_id, position_title,department, salary,
RANK() OVER(PARTITION BY department ORDER BY salary DESC)
FROM employees
NATURAL LEFT JOIN departments
```

14. Write a query that returns only the top earner of each department including their emp_id, position_title, department and their salary.

```
SELECT * FROM(SELECT emp_id, position_title,department, salary,
RANK() OVER(PARTITION BY department ORDER BY salary DESC)
FROM employees
NATURAL LEFT JOIN departments) aWHERE rank=1
```

Day 14-15

Day 14:

User Defined Functions:

Syntax:

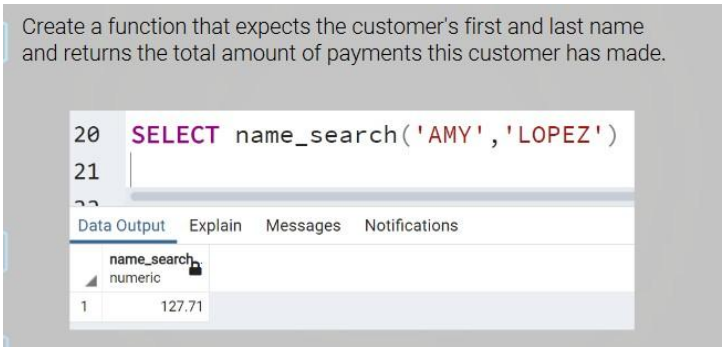
```
CREATE FUNCTION <function_name> (param1, param2,…)
RETURNS return_datatype
LANGUAGE plpgsql [sql|c|…]
AS
$$
DECLARE
<variable declaration>;
BEGIN
<function_definition>;
END;
$$
```

Example:

```
Create function count_rr(min_r decimal(4,2),max_r decimal(4,2))
returns int
language plpgsql
as
$$
Declare
movie_count int;
begin
select count(*)
into movie_count
from film
where rental_rate between min_r and max_r;
return movie_count;
end;
$$
;
select count_rr(0,3);
```

This gives no of payments between 0 and 3

Challenge:



```
create function name_search(
   first_n varchar(10),last_n varchar(10))
returns float
language plpgsql
as
$$
declare
total_amount float;
begin
select sum(amount)
into total_amount
from payment p right join
customer c on p.customer_id=c.customer_id
where first_name=first_n and last_name=last_n;
return total_amount;
end;
$$
;
select name_search('AMY','LOPEZ');
```

Transactions:

It can be done using three types
begin transaction;
begin work;
begin;
If the transaction is not commited then in other sessions changes won't be updated

Syntax:

```
BEGIN;
OPERATION1;
OPERATION2;
COMMIT;
```

Example:

```
begin;
update acc_balance
set amount =amount-100
where id=1;
commit;
```

Challenge:

The two employees Miller McQuarter and Christalle McKenny have agreed to swap their positions incl. their salary.

emp_id [PK] integer	first_name text	last_name text	position_title text	salary numeric (8,2)
1	Morrie	Conaboy	CTO	21268.94
2	Miller	McQuarter	Head of BI	14614.00
3	Christalle	McKenny	Head of Sales	12587.00

```
begin;
update employees
set
position_title='Head of Sales'
where emp_id=2;
update employees
set
position_title='Head of BI'
where emp_id=3;
update employees
set
salary=12587.00
where emp_id=2;
update employees
set
salary=14614.00
where emp_id=3;
commit;
```

Roll back:

If we did a mistake in updates then we can use roll back before commiting.
When we roll back it delete all the transaction

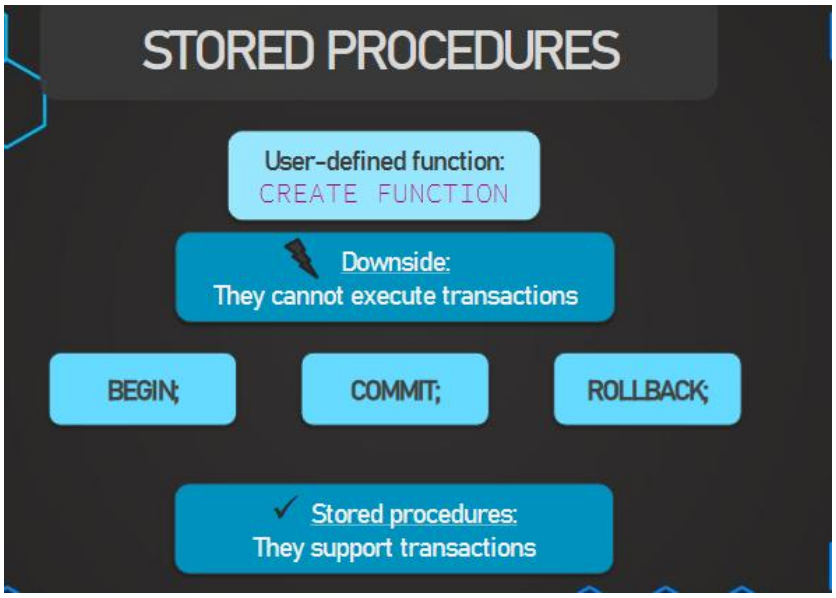
```
BEGIN;
OPERATION1;
OPERATION2;
OPERATION3;
OPERATION4;
ROLLBACK;
COMMIT;
```

Save Point:

we can use save point to save some of the operations in the transaction. This can be useful when we use rollback savepoint to save the correct operations from deleting.

```
BEGIN;
OPERATION1;
OPERATION2;
SAVEPOINT op2;
OPERATION3;
OPERATION4;
ROLLBACK TO SAVEPOINT op2;
COMMIT;
```

Stored procedure:



Syntax:

Creating procedure:

```
CREATE PROCEDURE <procedure_name> (param1, param2,…)
LANGUAGE plpgsql [sql|c|…]
AS
$$ DECL
ARE
<variable declaration>
BEGIN
<procedure_definition>
END;
$$
```

Calling a procedure:

```
CALL <store_procedure_name> (param1, param2,…);
```

Example:

```
create or replace procedure sp_transfer
(tr_amount int,sender int,recipient int)
language plpgsql
as
$$
 begin
update acc_balance
set amount=amount+tr_amount
where id=recipient;
update acc_balance
set amount=amount-tr_amount
```

```
where id=sender;
commit;
ENd;
$$

call sp_transfer (500,1,2)
```

This adds 500 to account with id 2 and subtracts from id1

Challenge:

Create a stored procedure called emp_swap that accepts two parameters emp1 and emp2 as input and swaps the two employees' position and salary

Test the stored procedure with emp_id 2 and 3.

emp_id [PK] integer	first_name text	last_name text	position_title text	salary numeric (8,2)
1	Morrie	Conaboy	CTO	21268.94
2	Miller	McQuarter	Head of BI	14614.00
3	Christalle	McKenny	Head of Sales	12587.00

```
create procedure emp_swap
(emp1 int,emp2 int)
language plpgsql
as
$$ decl
are
salary1  decimal(8,2);
salary2  decimal(8,2);
title1 text;
title2 text;
begin
--saving salaries,position in variables
select salary
into salary1
from employees where emp_id=emp1;
select salary
into salary2
from employees where emp_id=emp2;
select position_title
into title1
from employees where emp_id=emp1;
select position_title
into title2
from employees where emp_id=emp2;
--changing title of emp1 to title of emp2
update employees
```

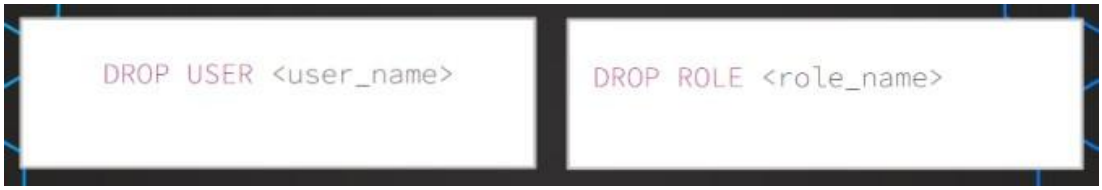
```
set position_title=title2
where emp_id=emp1;
--changing title of emp2 to title of emp1
update employees
set position_title=title1
where emp_id=emp2;
--changing salary of emp1 to salary of emp2
update employees
set salary=salary2
where emp_id=emp1;
--changing salary of emp2 to salary of emp1
update employees
set salary=salary1
where emp_id=emp2;
end;
$$

call emp_swap(2,3)

select * from employees
```

Day 15:

CREATE USER:
both create user or create role both are same



We intially will be postgre super user
To change the user right click on postrgre then properties and then under connection we can change username

new users cannot edit or use the tables as they have no permissions.
But they can create their own table and edit it or delete it etc

Privileges:

Example:

GRANT SELECT
ON customer
TO nikolai

Example:
create user ria
with password 'ria123';

create user mike
with password 'mike123';

create role read_only;
create role read_update;

grant usage on schema public to
read_only;

grant select on all
tables in schema public to read_only;
grant read_only to mike

grant read_only to read_update

grant all on all tables in schema public

to read_update

revoke delete,insert on all tables in schema public
from read_update

grant read_update to ria

drop role mike;
drop role read_update;

We can drop users easily
But while dropping roles if a user depends on the role it can't be dropped.
In above example ria is dependent on read_update so it cant be dropped
To drop it we should drop all it owns and then we can drop read_update

Drop owned by read_update
drop role read_update

Challenge:

In this challenge you need to create a user, a role and add privileges.
Your tasks are the following:

1. Create the user mia with password 'mia123'
create user mia
with password 'mia123'

2. Create the role analyst_emp;
create role analyst_emp;

3. Grant SELECT on all tables in the public schema to that role.
grant select on all tables in schema public
to analyst_emp

4. Grant INSERT and UPDATE on the employees table to that role.
grant insert,update on employees
to analyst_emp

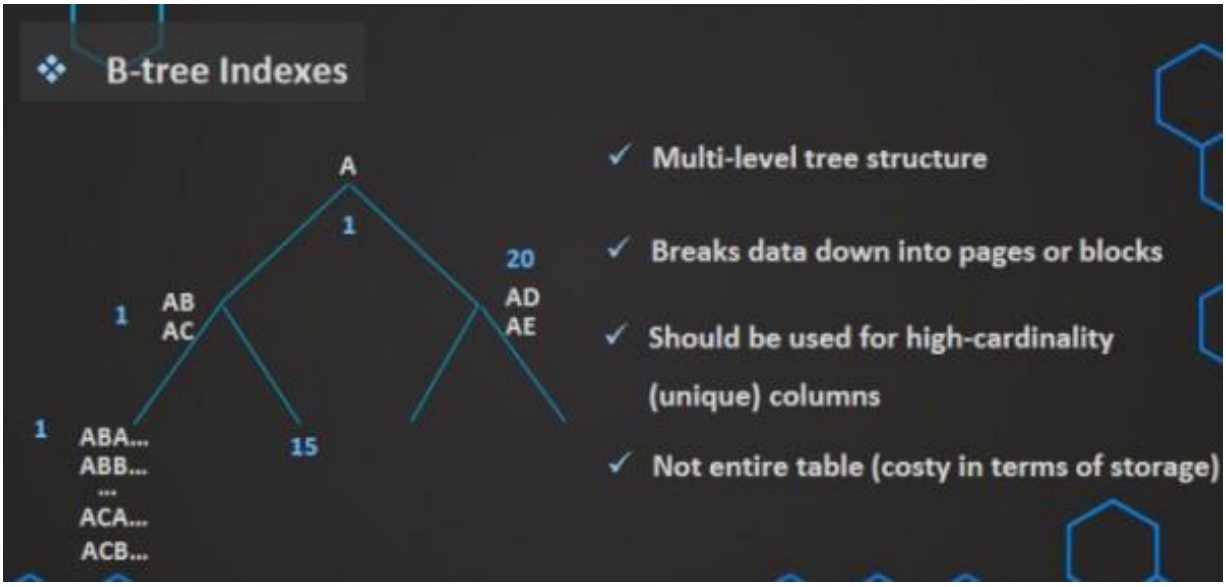
5. Add the permission to create databases to that role.
ALTER ROLE analyst_emp CREATEDB;

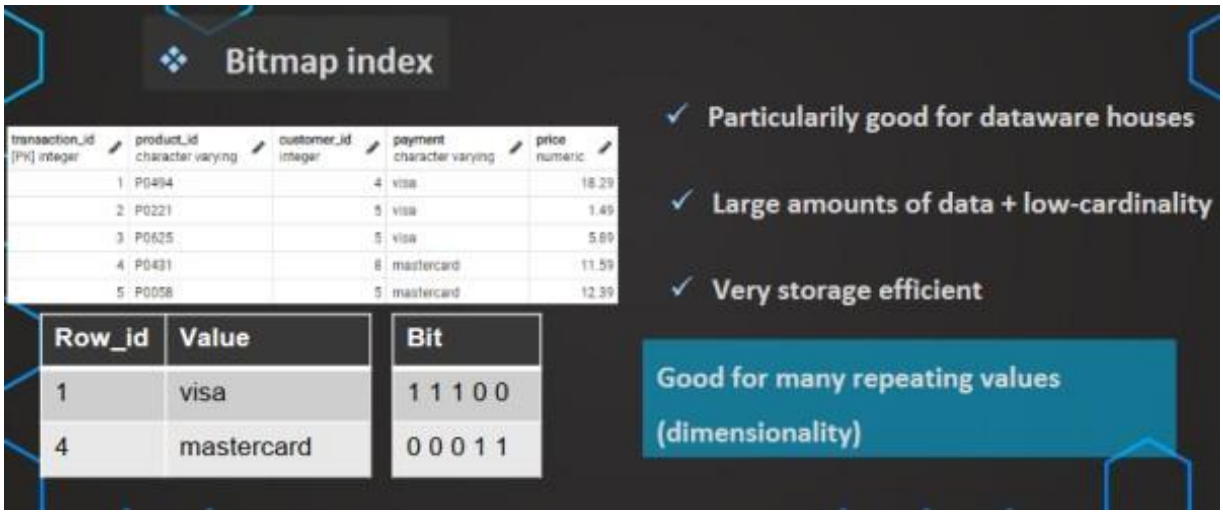
6. Assign that role to mia and test the privileges with that user.
GRANT analyst_emp TO mia;

INDEXES:

Indexes helps to point the data.
Indexes help to make data reads faster

Different types of indexes
1. B-tree indexes
2. Bitmap indexes





Guidelines to index a column:

Should we put index on every column?
No! They come with a cost!
Only when necessary!
Avoid full table reads
Small tables do not require indexes

On which columns?
1. Large tables
2. Columns that are used as filters

To find the column that is filter
we use **EXPLAIN AND ANALYZE** button on the query.In the graphical representation we click on the table we get a box showing the filter.

Creating indexes:

```
CREATE INDEX index_name
ON table_name
(
column_name1,
column_name2
);
```

```
select (select avg(amount)from payment p2
where p2.rental_id=p1.rental_id
)
from payment p1;
```

This query takes long time to run .But when we create index it runs easily in less time.
When we use explain and analyze we see that filter is rental_id.So create index on it.

```
create index index_rental_id_payment
on payment
(rental_id)
```

After running the above query , run avg amount query ,it now gives output faster