DB overview

* Systems that store & organize data
* Useful dealing with large amounts of data

PostgreSQL

select district, email from customer c

join address a

on c.address\_id = a.address\_id where district = 'California'

select title, first\_name, last\_name from film f

join film\_actor fa

on f.film\_id = fa.film\_id

join actor a

on fa.actor\_id = a.actor\_id

where a.first\_name = 'Nick'

and last\_name = 'Wahlberg'

show TIMEZONE

select now()

select timeofday()

select current\_date

select current\_time

select extract(day from payment\_date) from payment

select extract(quarter from payment\_date) from payment

select age(payment\_date) from payment

select to\_char(payment\_date, 'MONTH - YYYY') from payment

select distinct(to\_char(payment\_date, 'Month')) from payment

select count(payment\_id) as dow from payment

where extract(dow from payment\_date) = 1

select round(rental\_rate/replacement\_cost,2)\*100 as percent from film

select rental\_rate \* 2 as increased\_rentalrate from film

**Case**

select customer\_id,

case

when (customer\_id < 100) then 'Premium'

when (customer\_id between 100 and 200) then 'Plus'

else 'Normal'

end as customer\_class

from customer

select customer\_id,

case customer\_id

when 2 then 'Winner'

when 5 then 'Second'

else 'Normal'

end as raffle\_results

from customer

**3.**

select

sum(

case rental\_rate

when 0.99 then 1

else 0

end

) as total

from film

4.

select

sum(

case rental\_rate

when 0.99 then 1

else 0

end

) as normal

sum(

case rental\_rate

when 2.99 then 1

else 0

end

) as plus

sum(

case rental\_rate

when 4.99 then 1

else 0

end

) as premium

from film

**Self Join**

select f1.title, f2.title, f1.length from film f1

join film f2

on f1.film\_id != f2.film\_id

and f1.length = f2.length

**Sub Queries**

select title, rental\_rate from film

where rental\_rate > (select avg(rental\_rate) from film)

**In**

select film\_id, title from film

where film\_id in

(select i.film\_id from rental r

join inventory i

on i.inventory\_id = r.inventory\_id

where return\_date

between '2005-05-29' and '2005-05-30')

**exists**

select first\_name, last\_name from customer c

where exists

(select \* from payment p where p.customer\_id = c.customer\_id and amount> 11)

**String operations**

**Coalesce –**

Select item,(price – coalesce(discount, 0)) as final from table

**Cast** – converst from one data type to other

**General sql syntax**

select cast('5' as integer)

**Postgresql syntax**

Select ‘5’::integer

select length(cast(inventory\_id as varchar)) from rental

**NULLIf**

Takes two arguments and returns null if both are equal

**View**

Is a db object that is stored query

View is kind of virtual table – it stores query instead of it’s results, whenever view is called it’s executed and results are fetched as if it’s from table

**Syntax**

Create or replace view viewname as query

or

Create view viewname as query

**Example**

Create view customer\_info as

Select \* from customer

Join address

on address.adress\_id = customer.address\_id

after this we can directly perform select from customer\_info

**Drop** view if exists view name

**Alter** **view** view\_name **rename to** new\_view\_name

**Import and export**

Allows to import data from csv file to existing table

Import doesn’t create table it assumes that table is already created

create table test1(

Id integer,

Name varchar(30),

Age integer

)

select \* from test1

Views

Sp

Index

Cursors

Clusters

PL/SQL