Storage Layer Assignment

**Creating table queries**

create table authors (

id Serial,

name varchar(50)

)

create table authors (

id Serial,

name varchar(50)

)

create table reviews (

id Serial,

rating integer,

reviewer\_id integer,

book\_id integer

)

1. select name, title

from authors

Natural join books

1. select name, title

from authors

full join books on authors.id = books.author\_id

1. select name, title, rating

from authors

join books on books.id = authors.id

join reviews on reviewer\_id = authors.id

where books.author\_id = reviews.id

1. select author\_id, count(id) as "Number of Books"

from books

group by author\_id

order by author\_id asc

1. select a.name,count(b.author\_id)

from books as b

right join authors as a

on a.id=b.author\_id

group by name

order by count(b.author\_id) desc

**Phones queries**

1. select manufacturer, (price \* units\_sold) as "Total\_Revenue"

from phones

where (price \* units\_sold) > 200000

1. SELECT name, price

FROM phones

WHERE price NOT IN (SELECT Max (price)

FROM phones)

order by price desc

LIMIT 2;

1. SELECT manufacturer, count(phones)

from phones

WHERE price < 170

GROUP BY manufacturer

having count(phones)>2;

1. select name,price, price::float / (select SUM(price) from phones) as sum\_price

from phones

1. select name, price

from phones

WHERE units\_sold > 5000

1. select name, manufacturer

from phones

where manufacturer IN ('Apple', 'Samsung')

1. select name, (price \* units\_sold) as total\_revenue

from phones

where (price \* units\_sold) > 100000

**Part 3**

1.

select (select count(paid)

from orders

where paid = true) ,(select COUNT(paid) from orders where paid = false)

2.

select first\_name, last\_name,paid

from users

NATURAL join orders

**UDEMY QUESTIONS**

**Section 24 Questions ‘Your turn’**

**126.**

CREATE OR REPLACE FUNCTION set\_employee\_default() returns void AS $$

UPDATE employees

SET photopath = 'http://accweb/employees/default.bmp'

WHERE photopath IS NULL;

$$ LANGUAGE SQL

**127.**

CREATE OR REPLACE FUNCTION biggestOrder() RETURNS

double precision AS

$$

SELECT MAX(amount) FROM (SELECT SUM (unitprice\*quantity) as amount, orderid

FROM order\_details

GROUP BY orderid) as totals

$$ LANGUAGE SQL

**128.**

CREATE OR REPLACE FUNCTION mostOrderedProduct(customerid ) returns varchar(50) AS

$$

SELECT productname from products where productid IN

(SELECT sum (quantity) as total\_ordered, productid

FROM order\_details

NATURAL JOIN orders

where customerid = $1

GROUP BY productid

ORDER BY total\_ordered DESC LIMIT 1) as products\_ordered

$$ LANGUAGE SQL

**129.**

CREATE OR REPLACE full\_name(employees) RETURNS varchar(70)

AS

$$

SELECT $1.title || ' ' || $1.firstname || ' '

$1.lastname;

$$ LANGUAGE SQL;

**130.**

CREATE OR REPLACE FUNCTION highest\_inv() RETURNS products as

$$

SELECT \* FROM products

ORDER BY (unitprice\*unitsinstock)

DESC

LIMIT 1;

$$ LANGUAGE SQL;

**131.**

CREATE OR REPLACE FUNCTION square\_cube(IN x int, OUT square int, OUT cube int) AS

$$

SELECT x\*x, x\*x\*x

$$ LANGUAGE SQL;

**132.**

CREATE OR REPLACE FUNCTION square\_cube(IN x int DEFAULT 5, OUT square int, OUT cube int) AS

$$

SELECT x\*x, x\*x\*x

$$ LANGUAGE SQL;

**133.**

select productname, companyname

FROM highest\_inventory as highestInv

JOIN suppliers on highestInv.supplierid on supplier.supplierid;

**134a)**

CREATE OR REPLACE FUNCTION suppliers\_to\_reorder\_from()

RETURN SETOF suppliers AS

$$

SELECT \* FROM suppliers

WHERE supplierid IN ( SELECT supplierid FROM products

WHERE unitsonorder + unitsonstock < reorderlevel

)

$$ LANGUAGE SQL;

**134b)**

CREATE OR REPLACE FUNCTION excess\_inventory\_level(percent numeric)

RETURNS TABLE (excess int, productid smallint, productname varchar(40))

AS

$$

SELECT CEIL(unitsonorder + unitsinstock - (reorderlevel + percent/100))::int,productid,productname

FROM products

$$ LANGUAGE SQL

**135)**

CREATE OR REPLACE PROCEDURE change\_supplier\_prices(supplierid smallint, amount real)

AS

$$

UPDATE products

SET unitprice = unitprice + amount

WHERE supplierid = $1

$$ LANGUAGE SQL

**UDEMY QUESTIONS**

**Section 27 Questions ‘Your turn’**

**147.**

CREATE OR REPLACE FUNCTION biggest\_order() RETURNS double precision as

$$

BEGIN

RETURN MAX(amount) FROM

(select SUM(unitprice \* quantity) as amount, orderid FROM

order\_details

GROUP BY orderid) as totals;

END;

$$ LANGUAGE plpgsql;

**148.**

CREATE OR REPLACE FUNCTION square\_cube(IN x int, OUT square int, OUT cube int)

AS

$$

BEGIN

square := x\*x;

cube := x\*x\*x;

RETURN;

END;

$$ LANGUAGE plpgsql

149.

**151.**

CREATE OR REPLACE FUNCTION averageSquare() RETURNS double precision AS

$$

DECLARE

square\_total int :=0;

total\_count int :=0;

BEGIN

FOR product in SELECT FROM products LOOP

total\_count := total\_count + 1;

square\_total := square\_total + (product.unitprice \* 2)

END LOOP;

RETURN square\_total / total\_count;

END

$$ LANGUAGE plpgsql;

**152.**

CREATE OR REPLACE FUNCTION timeofyear (date\_to\_check timestamp) RETURNS text

AS

$$

DECLARE

monthOfYear int := EXTRACT (MONTH FROM date\_to\_check);

BEGIN

IF monthOfYear >= 3 AND monthOfYear <=5

THEN

RETURN 'Spring';

ELSEIF monthOfYear >= 6 AND monthOfYear <=8

THEN

RETURN 'Summer';

ELSEIF monthOfYear >= 9 AND monthOfYear <=11

THEN

RETURN 'Fall';

ELSE RETURN 'Winter';

$$ LANGUAGE plpgsql;