Important Links:

WhatsApp: https://chat.whatsapp.com/invite/FofcmptRtuRLdLgyTx6bsO

Connect with Instructor: https://www.linkedin.com/in/saurabhkango/

If you are facing any challenges or issues regarding the lectures, contact support@scaler.com

For all queries related to the dashboard, recordings, assessments and material access - support link: https://support.scaler.com/hc/en-us

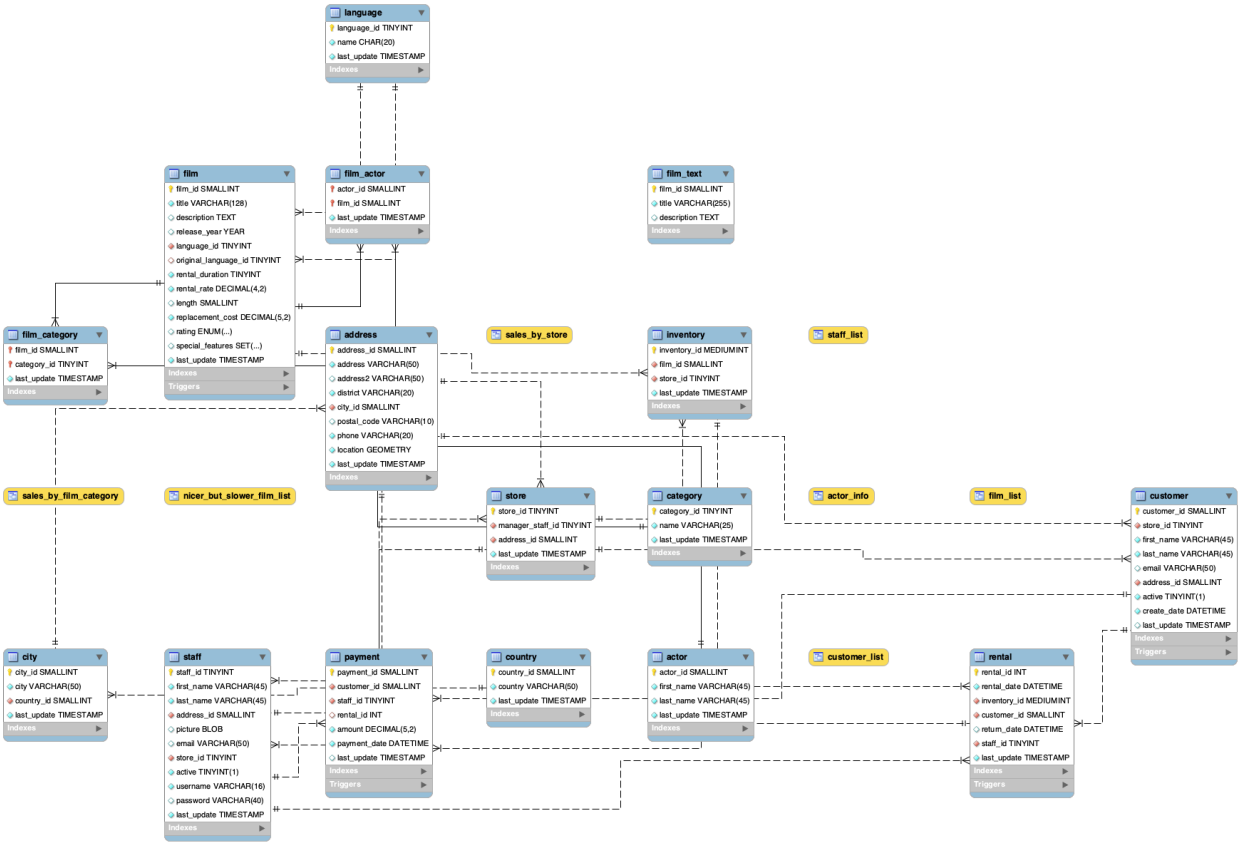
MySQL - MySQL Installation and Setup.pdf

Don't forget the password.

Data:
https://dev.mysql.com/doc/index-other.html

https://drive.google.com/drive/folders/17jrBlXgzZAyJ-GvOJ1sAb0gPk8rHsy26

ER diagram:



Notes:

Concept DE

```
use  scaler_demo ;

CREATE TABLE departments (
    deptID INT AUTO_INCREMENT PRIMARY KEY,
    deptName VARCHAR(50) NOT NULL
);

CREATE TABLE employees (
    employeeID INT AUTO_INCREMENT PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50),
    deptID INT,
```

```
    FOREIGN KEY (deptID) REFERENCES departments(deptID)
);
INSERT INTO departments (deptName)
VALUES ("HR"), ("TECH"), ("PRODUCT"), ("MARKETING");

INSERT INTO employees (firstName, lastName, deptID)
VALUES
("Mohit", "Sharma", 1),
("Abdul", "Ahad", 2),
("Naman", "Bhalla", 3),
("Rahul", "Sharma", 1);

--
Select * from departments ;
Select * from employees ;

DELETE FROM departments WHERE deptID = 3;

DROP TABLE employees;
DROP TABLE departments;

CREATE TABLE departments (
    deptID INT AUTO_INCREMENT PRIMARY KEY,
    deptName VARCHAR(50) NOT NULL
);

CREATE TABLE employees (
    employeeID INT AUTO_INCREMENT PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50),
    deptID INT,
    FOREIGN KEY (deptID) REFERENCES departments(deptID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);


DELETE FROM departments WHERE deptID = 3;

UPDATE departments
SET deptID = 5
WHERE deptID = 1;
```

```sql
CREATE TABLE departments (
    deptID INT AUTO_INCREMENT PRIMARY KEY,
    deptName VARCHAR(50) NOT NULL
);

CREATE TABLE employees (
    employeeID INT AUTO_INCREMENT PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50),
    deptID INT NULL,   -- MUST allow NULL
    FOREIGN KEY (deptID) REFERENCES departments(deptID)
        ON DELETE SET NULL
        ON UPDATE SET NULL
);
```

```sql
use sakila;
Select f.film_id,
          f.title,
     l.name as language_name
from film f
left join language L
on l.language_id = f.language_id;

Select f.film_id,
          f.title,
     l.name as language_name
from film f
inner join language L
on l.language_id = f.language_id;


Select f.title,
         count(r.rental_id) as total_rents
from film F
left join inventory i
on f.film_id = i.film_id

left join rental r
on r. inventory_id  = i.inventory_id
group by f.title
```

```
order by total_rents desc



Select f.title,
          count(r.rental_id) as total_rents
from film F
 join inventory i
on f.film_id = i.film_id

 join rental r
on r. inventory_id  = i.inventory_id
group by f.title
order by total_rents desc
```

Pivots 👍

```
Select sum(case when rating ='PG' then 1 else 0 end) as pg_movies,
            sum(case when rating ='G' then 1 else 0 end) as g_movies,
      sum(case when rating ='PG-13' then 1 else 0 end) as pg13_movies,
      sum(case when rating ='R' then 1 else 0 end) as R_movies,
      sum(case when rating ='NC-17' then 1 else 0 end) as NC_movies
from film;



-- Columns to Rows:
Select film_id, 'length' as kpi, length as value
from film
union all
Select film_id, 'rental_rate' as kpi, rental_rate as value
from film
union all
Select film_id, 'replacement_cost' as kpi, replacement_cost as value
from film
order by film_id
```

```
-- CTE/ Sub Query
Select *
from
(Select Rating,
            avg(rental_rate) as avg_rental_rate
from film
group by 1) as a
where avg_rental_rate > 3

-- CTE
with final as
(Select Rating,
            avg(rental_rate) as avg_rental_rate,
      max(rental_rate) as max_rental_rate
from film
group by 1)

Select *
from final
where avg_rental_rate > 0.58 * max_rental_rate

-- Find the ratings where avg_rental_rate > max(avg_rental_rate)
Select *
from
(Select Rating,
            avg(rental_rate) as avg_rental_rate,
      max(rental_rate) as max_rental_rate
from film
group by 1) as a
where avg_rental_rate > (Select max(rental_rate) from (Select Rating,

          avg(rental_rate) as avg_rental_rate,

          max(rental_rate) as max_rental_rate

          from film
                                                                    group by 1)

with final as
(Select Rating,
            avg(rental_rate) as avg_rental_rate,
      max(rental_rate) as max_rental_rate
from film
group by 1)
```

```
Select *
from final
where avg_rental_rate >= (Select max(avg_rental_rate) from final)

-- Rating where avg_rental_rate) >= max(avg_rental_rate)
Select *
from
(Select Rating,
            avg(rental_rate) as avg_rental_rate,
      max(rental_rate) as max_rental_rate
from film
group by 1) a
where avg_rental_rate >= (Select max(avg_rental_rate) from a)

with rental_rate as (
Select rating,
            avg(rental_rate) as avg_rental_rate
from film
group by 1),

replacement_cost as (
Select rating,
            avg(replacement_cost) as avg_replacement_cost
from film
group by 1)

Select rr.rating,  rc.avg_replacement_cost, rr.avg_rental_rate
from replacement_cost rc
inner join rental_rate rr
on rc.rating = rr.rating


-- So we want an actual language, but we have language_id, how can we get language for
every movie ?
use sakila;
Select f.film_id,
            f.title,
      l.name as language_name
from film f
left join language L
on l.language_id = f.language_id;

Select f.film_id,
```

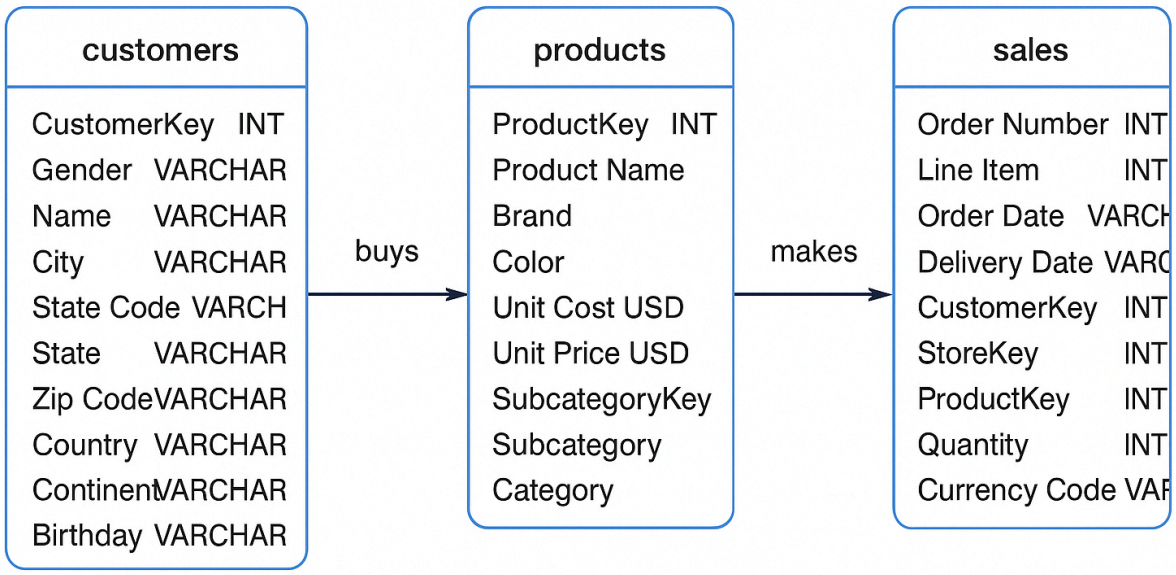
```
            f.title,
      l.name as language_name
from film f
inner join language L
on l.language_id = f.language_id;


Select f.title,
          count(r.rental_id) as total_rents
from film F
left join inventory i
on f.film_id = i.film_id

left join rental r
on r. inventory_id  = i.inventory_id
group by f.title
order by total_rents desc



Select f.title,
          count(r.rental_id) as total_rents
from film F
 join inventory i
on f.film_id = i.film_id

 join rental r
on r. inventory_id  = i.inventory_id
group by f.title
order by total_rents desc
-- Rows to columns

Select sum(case when rating ='PG' then 1 else 0 end) as pg_movies,
            sum(case when rating ='G' then 1 else 0 end) as g_movies,
      sum(case when rating ='PG-13' then 1 else 0 end) as pg13_movies,
      sum(case when rating ='R' then 1 else 0 end) as R_movies,
      sum(case when rating ='NC-17' then 1 else 0 end) as NC_movies
from film;

-- Columns to Rows:
Select film_id, 'length' as kpi, length as value
from film
union all
Select film_id, 'rental_rate' as kpi, rental_rate as value
```

```
from film
union all
Select film_id, 'replacement_cost' as kpi, replacement_cost as value
from film
order by film_id


Select film_id, length, rental_rate,replacement_cost
from film


-- CTE/ Sub Query
Select *
from
(Select Rating,
            avg(rental_rate) as avg_rental_rate
from film
group by 1) as a
where avg_rental_rate > 3

-- CTE
with final as
(Select Rating,
            avg(rental_rate) as avg_rental_rate,
      max(rental_rate) as max_rental_rate
from film
group by 1)

Select *
from final
where avg_rental_rate > 0.58 * max_rental_rate

-- Find the ratings where avg_rental_rate > max(avg_rental_rate)
Select *
from
(Select Rating,
            avg(rental_rate) as avg_rental_rate,
      max(rental_rate) as max_rental_rate
from film
group by 1) as a
where avg_rental_rate > (Select max(rental_rate) from (Select Rating,

          avg(rental_rate) as avg_rental_rate,
```

```
        max(rental_rate) as max_rental_rate

        from film
                                                                group by 1)

with final as
(Select Rating,
            avg(rental_rate) as avg_rental_rate,
      max(rental_rate) as max_rental_rate
from film
group by 1)

Select *
from final
where avg_rental_rate >= (Select max(avg_rental_rate) from final)

-- Rating where avg_rental_rate) >= max(avg_rental_rate)
Select *
from
(Select Rating,
            avg(rental_rate) as avg_rental_rate,
      max(rental_rate) as max_rental_rate
from film
group by 1) a
where avg_rental_rate >= (Select max(avg_rental_rate) from a)

with rental_rate as (
Select rating,
            avg(rental_rate) as avg_rental_rate
from film
group by 1),

replacement_cost as (
Select rating,
            avg(replacement_cost) as avg_replacement_cost
from film
group by 1)

Select rr.rating,  rc.avg_replacement_cost, rr.avg_rental_rate
from replacement_cost rc
inner join rental_rate rr
on rc.rating = rr.rating

Select *, avg(rental_rate) as avg_rental_rate
```

```
from film
group by 1,2,3,4,5,6,7,8,9,10,11,12,13

Select
            avg(rental_rate) as avg_rental_rate
from film
-- FO
Select *, avg(rental_rate) over() as overall_avg
from film



Select rating,
            avg(rental_rate) as avg_rental_rate
from film
group by 1

-- FOP
Select *, avg(rental_rate) over(partition by rating) as overall_avg
from film

-- Running calc - FOO
Select film_id,replacement_cost,
            sum(replacement_cost) over( order by film_id) as running_total
from film

Select film_id,replacement_cost,
           avg(replacement_cost) over( order by film_id desc) as running_total
from film

Select film_id,replacement_cost,rating,
           sum(replacement_cost) over( partition by rating order by film_id desc) as running_total
from film


Select film_id,replacement_cost,rating,
             rank()over(order by replacement_cost desc) as rnk,
       dense_rank()over(order by replacement_cost desc) as drnk,
        row_number()over(order by replacement_cost desc) as rnum
from film


Select film_id,replacement_cost,rating,
```

```
        dense_rank()over(partition by rating order by replacement_cost desc) as drnk
from film

with final as (
Select film_id,replacement_cost,rating,

        dense_rank()over(partition by rating order by replacement_cost desc) as drnk
from film)
Select *
from final
where drnk =2
```




```
create database dmart;
use dmart ;

-- % MoM increase/desc in qty sold
with final_data as(
Select month(`Order Date`) as month_dt, sum(quantity) as total_qty
from sales
group by 1
)
```

```
Select *, lead(total_qty,2) over( order by month_dt) as next_mnth_Sales,
                    100* (lead(total_qty,1) over( order by month_dt)- total_qty)/total_qty as mom
from final_data

Select CustomerKey, `Order Date` ,
         first_value(`Order Date`) over(partition by CustomerKey order by  `Order Date` ) as fv,
   nth_value(`Order Date`,2) over(partition by CustomerKey order by  `Order Date` ) as sv
from  sales


with final_data as(
Select month(`Order Date`) as month_dt, sum(quantity) as total_qty
from sales
group by 1
)

Select *, sum(Total_qty) over(order by month_dt rows between 2 preceding and current row) as
run_t
from final_data

with final_data as(
Select month(`Order Date`) as month_dt, sum(quantity) as total_qty
from sales
group by 1
)

Select *, sum(Total_qty) over(order by month_dt rows between current row and 2 following) as
run_t
from final_data

Select CustomerKey, `Order Date` ,
         first_value(`Order Date`) over(partition by CustomerKey order by  `Order Date` ) as fv,
   nth_value(`Order Date`,2) over(partition by CustomerKey order by  `Order Date`
                                                                    rows between unbounded
preceding and unbounded following) as sv
from  sales

-- Calculate the total revenue generated in 2019 and 2020.
-- Compare the revenue figures to identify the percentage decline
with final as (
Select year(`Order Date`) as yr_dt,
            sum(s.Quantity* p.`Unit Price USD`) as revenue
from sales s
left join products p
```

```
on p.ProductKey = s.ProductKey
where year(`Order Date`) in (2019,2020)
group by 1)

Select *,
        round(100* (lead(revenue,1)over(order by yr_dt)-revenue)/revenue,1) as perc_yoy
from final

-- Calculate the monthly revenue trend for 2020

Select month(`Order Date`) as month_dt,
        round(sum(s.Quantity* p.`Unit Price USD`),2) as revenue
from sales s
left join products p
on p.ProductKey = s.ProductKey
where year(`Order Date`) in (2020)
group by 1

-- Q3. Identify the top 10 customers who made the highest number of orders in 2020. - Easy

Select c.CustomerKey,Name,
        count(`Order Number`) as totalorders
from sales s
left join customers c
on c.CustomerKey =s.CustomerKey
where year(`Order Date`) in (2020)
group by 1,2
order by 3 desc
limit 10;


-- Q4. Find the customer with the highest total spending in each state. - Medium
with final as (
Select c.CustomerKey,
                Name,
    state,
    round(sum(s.Quantity* p.`Unit Price USD`),2) as revenue,
    dense_rank() over( partition by state order by sum(s.Quantity* p.`Unit Price USD`) desc) as
rnk
from sales s
left join customers c
on c.CustomerKey =s.CustomerKey
left join products p
on p.ProductKey = s.ProductKey
```

```
group by 1,2,3)

Select CustomerKey,
                Name,
       state
from final
where rnk =1
```