Date

Expt. No. 27/5/23 Page No. 1

Structured Query Languages (SQL)

- How to check databases :

  show databases ;

- How to create database :

  Create database < database name > ;

- How to delete database :

  Drop database < database name > ;

- How to check current database :

  (i) Use the database

  use < database name > ;

  (ii) select database ( ) ;

Teacher's Signature

Students Table

Fields →

Records ↓

Data Types

1. Numeric
   - INT
2. String
   - VARCHAR
   - eg VARCHAR(25)
     - ↳ 25 bytes in each row.
     - ↳ 'Vikas'.
3. Date & Time
   - DATE
   - DATETIME
     - ↳ YYYYMMDD.

- How to creating Tables:

```
Create Table <Table name>
    ( Column name   data type,
      Column name   data type,
      .....         .....
    );
```

- How to show Tables:

```
show Tables;
```

→ Desc < Table name> ;

→ Show columns from < Tables name >;

- How to delete Table:

```
Drop table < Table name >;
```

- Insert data into Table:

```
Create table students ( id int, name varchar(55),
                        class int );

show tables;
```

① Insert into <table name> (Column 1, Column 2, -------)

Values (value 1, value 2, -------);

② Insert into < Table name> Values (value 1, value 2, -----);

- Show Data of Table:

```
Select * from < Table name> ;
```

- How to show warnings:

```
Show warnings;
```

* **Null** → The value is Not Known.

↳ Null doesn't mean zero.

```
Create table students ( id int Not Null,
                        name varchar(55) Not Null,
                        class int Not Null );
```

→ Default value Table:

```
Create Table students ( id int not null
                              default 1,
                        name varchar(55) not null
                              default 'unnamed',
                        class int not null default 'no class');
```

- Alter table statement in MySQL :

① Alter table <table name> drop column <column name>;

② ↳ alter table students add school varchar(55);

* Primary Key:

→ Must contain UNIQUE values, cann't contain NULL values.

→ Constraint uniquely identifies each record in a table.

- Auto_increment and Primary key :

Create table loki (sno int not null auto-increment primary key,

name varchar(55) not null, class int not null);

CRUD

Create	Read	Update	Delete
1	2	3	4

1. Create table students (id int not null auto_incremant primary key, fname varchar(55), lname varchar(55), age int not null);

2. Desc <table name>;

Desc students;

→ Insert the data into table.

→ Select * from students;

2 ☆ **Read:**

Where clause in MySQL ☆

→ The where clause is used to filter records.

↳ Select * from students where age = 16;

<table name>

***

3 The UPDATE statement is used to modify existing record in a table.

<tablename>

→ UPDATE students set age = 30
where fname = 'ramesh';

4

<tablename>

Delete from students where fname = 'vikas';

→ Delete from students;

* The select Distinct statement is used to return only distinct or different values.

→ Select Distinct column1, column2, ... from <table name>;

eg. Select distinct fname from students;

select distinct fname, age from students;

* Order By :
Either increasing or decreasing.

↳ Select age from students order by age;

↳ Select distinct age from students order by age desc;

↳ Select age, fname from students order by age;

* Limit :

Select fname from students order by fname limit 3;

↳ only 3 fname will be there in output in the Ascending order.

* **Like operator:** is used in a where clause to search for a specified pattern in a column.

→ Select fname from students
where fname like '%as%';

*

COUNT()	no. of rows
AVG()	average value of numeric column
SUM()	Total sum of " "

↳ Select max(age) from students;

* String Function:

① MySQL CONCAT() Function :

→ The concat() function adds two or more expressions together.

CONCAT(expression1, expression2, ...)

↳ select CONCAT (fname, ' ', lname) from students as Fullname;

② REVERSE() Function:

→ reverse a string and returns the result.

Select reverse ('fname') from students;

select concat ('Vikas', reverse('Vikas')) as palindrom;

③ CHAR_LENGTH
LCASE()
UCASE()

Date time DATA TYPE :

Date format: YYYY-MM-DD
Time format: HH:MI:SS
Date time format: YYYY-MM-DD HH:MI:SS

Create table datestu (name varchar(55), dob date, dot time, ~~dot~~ bt date time);

☆ → All are in ' '

→

CUR DATE ()	Gives the current date:	YYYY-MM-DD
NOW()	Gives both Current date and time	YYYY-MM-DD HH:MI:SS
CUR TIME ()	Gives current time	HH:MI:SS

→ ' ' not in CURDATE(), CURTIME() and NOW(). only in manually.

* **Foreign Key:** Used to link two tables together.

→ Is a field in one table that refers to the primary key in another table.

→

```
Create table customers ( cid int not null auto_increment
                        primary key, cname varchar (55),
                        email varchar (100) );

Create table orders ( oid int not null auto_increment
                      primary key, odate date , cid int,
                      foreign key (cid) references customers (cid));
```

CUSTOMERS:

cid	cname	email

ORDERS:

oid	odate	cid

→ Insert data into both tables.

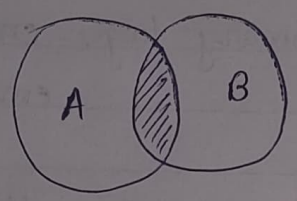
'cid' is ambiguous.

JOINS

→ A Join clause is used to combine rows from two or more tables, based on a related column b/w them.

(i) Inner Join:

Select records that have matching values in both tables.

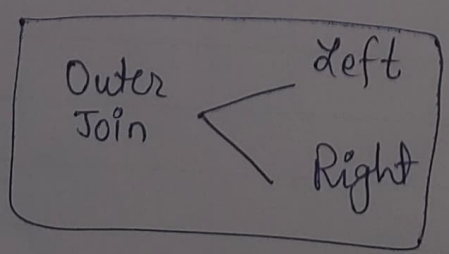


↳ Select * from customers, orders where customers.cid = orders.cid order by orderdate;

↳ Select * from customers inner join orders on customers.cid = orders.cid;

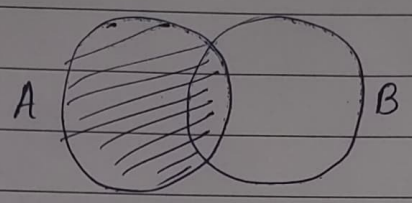
↳ Select cname, email, orderdate from customers inner join orders on customers.cid = orders.cid;

↳ Select cname, customer.cid from customers inner join orders on customers.cid = orders.cid;



(ii) Left Join:

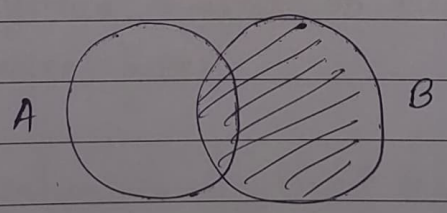
Returns all records from Left table (Table 1) and matched records from right table (Table 2).



↳ Select columnname from Table 1 Left join Table 2 on Table 1. column name = Table 2. columnname;

↳ Select * from Customers left join orders on customers. cid = orders. cid;

(iii) Right Join:



↳ Select columnname from Table 1 Right Join Table 2 on Table 1. columnname = Table 2. columnname;

↳ Select cname from ~~Table 1~~ Customers right Join orders on customers. cid = orders. cid

#END#

Teacher's Signature विकास चौधरी
31/5/23

27-7-23

* BIG 6 Elements of a SQL Select Statement:

Select
FROM
where
Group by
Having
Order by

→ Select ColumnName
from table Name
Where logical Condition
Group by ColumnName
Having logical Condition
Order by ColumnName

Select * from Table_name;

Select customer_id, rental_date from rental;

→ Select Distinct customer_id from rental;

Where Logical Condition clause

→ Select → From → Where

→ Using AND/OR statements within your where clause.

operators:

Between ⇒ A Range between Two values
Like ⇒ Matching a pattern like this
IN() ⇒ Equals one of these values
<> ⇒ Does not equal

Select customer_id, rental_id, amount, payment_date
from payment
Where amount = 0.99 ;

→ '2006-12-01'

customer_id IN (42, 53, 60, 75);

→ Select title, description from film where title like '_China' ;

Like Not Like

Group by columnName, othercolumnName

→ Select rating, count (film_id) from film Group by rating ;

-- /* comments.

Aliases allow you to assign a custom name to a field in your result set, using an AS statement.

→ Select rating, count(film_id), count (film_id) AS films_with_this_rating
from film
Group by rating;

→ Having use with Group by.

Select → from → Where → Group by → Having → Order by

Select * from rental;

Select * from inventory;

select customer_id, rental_date from rental;

Select Distinct rating from film;

Where → Clause

(optional)

↳ specifies criteria for filtering the records of result set.

Name or date in Columns ' ' 'YYYY-MM-DD'

→ Select from where Group by Having Order by

→ Use AND/OR statements with Where clause.

Select customer_id, rental_id, amount, payment_date
from payment Where amount = 0.99

Select customer_id, rental_id, amount, payment_date
from payment Where payment_date > '2006-12-01';

→ We can use AND/OR in where clause. This is powerful.

Select customer_id, rental_id, amount, payment_date
from payment Where customer_id = 5
OR customer_id = 11
OR customer_id = 29;

Select customer_id, rental_id, amount, payment_date
from payment where customer_id IN (5, 11, 29);

LIKE operator

Allow us to use pattern matching in logical operator

Like '% kas %'

(Capitalization Matter)

'Denise%' '% Johnson'

'% fancy%' '_kas'

→ NOT LIKE can also be used to filter out record.

Select title, description from film
Where description like '% Dentist %'

Select title, special_features from film
where special_features like '% Behind the Scenes'

Group by Clause

(optional)

↳ specifies how to group the data in result.

Select → from → Where → Group by → Having → Order by

→ Group by is great for comparing different segment of data.
(Similar to Pivot Table in Excel)

select rating, count(film_id) from film
Group by rating;

COMMENTS, ALIASES

↳ AS

Single line → --

Multiple line → /* */

Select rating, COUNT (film_id) AS films_with_this_rating
from film
Group by rating;

↳ Select rental_duration, count (title) from film
Group by rental_duration;

Select rating, rental_duration, count(film_id) AS count_of_films
from film Group by rating, rental_duration;

Aggregate functions:

Select rating, count (film_id) AS count of films,
Min (length) AS shortest_film, MAX(length) AS longest_film,
Avg (length) AS average_length_film
from film
Group by rating;

HAVING CLAUSE

(optional)

↳ Specifies group filtering criteria for filtering result.

Select → From → Where → Group by → Having → Order by

→ Having can only be used with Group by.

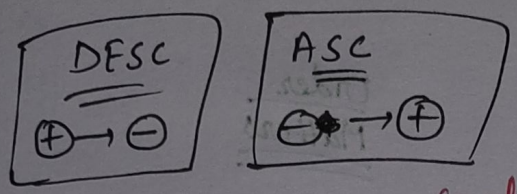
```
Select customer_id , count(*) As total_rental
from rental
Group by customer_id
Having count (*) >= 30;
```

```
Select customer_id, count (rental_rate) As rentals
from ~~film~~ rental
Group by customer_id
Having count(rental_rate) < 15;
```

ORDER BY

(Optional)

↳ specifies the order in which query result are displayed.



Select → from → where → ~~Group~~ Group by → Having → Order by
① ② ③ ④ ⑤ ⑥

```
Select customer_id, rental_id, amount, payment_date
from payment
Order by amount DESC;
```

```
Select customer_id, sum(amount) AS Total
from payment
Group by customer_id
Order by sum(amount) DESC;
```

→

CASE STATEMENT

→ Allow to process a series of if / then logical operator in a specific order.

Order Matters

```
Case
When logic 1  Then value 1
When logic 2  Then value 2
Else value 3
End
```

→ Select distinct length,
Case When length <60 then 'under 1 hr'
When length between 60 and 90 then '1-1.5 hr'
When length >90 then 'over 1.5 hr'
Else 'check logic'
End As length_bucket
From film;

→ Select Distinct title,
Case When rental_duration <=4 then 'rental_too_short'
When rental_rate >= 3.99 then 'too_expensive'
When rating IN('NC-17','R') then 'too_adult'
When length Not between 60 and 90 then 'too_short_or_too_long'
When description like '% shark %' then 'nope_has_sharks'
Else 'great_reco_for_my_niece'
End As fit_for_recommendation
from film;

```
Select store_id,
count ( case when active=1 then customer_id else null end) as active,
count (  "    "    "  =0  "    "    "    "    "    ) as inactive,
from customer
Group by store_id;
```

MID - COURSE PROJECT

Q.1. We will need a list of all staff member, including their first and last names, email, and the store identifications number where they work.

Sol:-
```
Select first_name, last_name, email, store_id
from staff;
```

Q.2. We will need separate counts of inventory items held at each of your two stores.

Sol:-
```
Select store_id,
count (inventory_id) As inventory_items
from inventory
Group by store_id;
```

Q.3. We will need a count of active customers for each of your stores. Separately, please.

Sol:-

```
Select store_id, count(customer_id) As active_customers
from customer
Where active =1
Group by store_id;
```

Q.4. In order to assess the liability of a data breach, we will need you to provide a count of all customers email stored in the database.

Sol:-

```
Select count (email) As emails
From customer;
```

Q.5 We are interested in how diverse your film offering is as a means of understanding how likely you are to keep customers engaged in future. Please provide a count of unique film titles you have in the inventory at each store and then provide a count of unique categories of films you provide.

Sol:- Select store_id,
count(Distinct film_id) As unique_films
from inventory
Group by store_id;

Select count(Distinct name) As unique_categories
from category;

Q.6. We would like to understand the replacement cost of your films. Please provide the replacement cost for the film that is least expensive to replace, the most expensive to replace and the average of all films you carry.

Sol:-
Select
min(replacement_cost) As least_expensive,
max(" ") As most_expensive,
Avg(" ") As average_replacement_cost
from film;

Q.7. We are interested in having you put payment monitoring systems and max. payment processing restrictions in place in order to min. the future risk of fraud by your staff. Please provide the avg ~~amount~~ payment you process, as well as max. payment.

Sol:-

```
select avg(amount) As avg_payment,
max(amount) As max_payment
from payment;
```

Q.8. We would like to better understand what your customer looks like. Please Provide a list of all customer identification values, with a count of rentals they have made all time, with your highest volume customer at the top of list.

Sol:-

```
Select customer_id,
count(rental_id) As no_of_rentals
from rentals
Group by customer_id
order by count(rental_id) DESC;
```

COMMON JOIN TYPE

✓ Inner Join

Returns records that exist in Both tables.

→ from leftTablename Inner Join right tablename

✓ Left Join

→ from leftTablename Left Join right Tablename

Right Join

→ From leftTablename Right Join rightTablename

Full outer Join

Returns all records from both tables, including non-matching records.

From leftTablename Full Join right Table name

UNION

Returns all data from one table, with all data from another table appended to the end.

Select from first Tablename UNION Select from secondTablename

INNER JOIN:

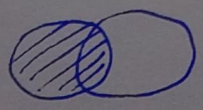
```
INNER JOIN right Table name
ON  left Table . column name = right Table . column name
```

```
from rental Inner Join customer
ON  rental. customer_id = customer . customer_id ;
```

```
Select inventory. inventory_id
from inventory inner Join rental
ON  inventory. inventory_id = rental. inventory_id ;
```

```
Select inventory_id, store_id, film.title, film. description
from inventory inner Join film
ON  inventory. film_id = film . film_id ;
```

Left JOIN :



Left Join right Tablename
ON left Tablename . column name = right Table . column name

```
Select Distinct inventory.inventory-id , rental. inventory-id
from inventory inner Join rental
ON inventory. inventory-id = rental. inventory-id ;
```