

MySQL Portfolio

Creating a database, showing the database and creating a table. There are columns with data types with a primary key and 'explain' function to view the fields in the output.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' list with 'hr7' selected. The main query editor contains the following SQL code:

```
1 CREATE DATABASE hr7;
2 SHOW DATABASES;
3 create table person(
4   person_id int not null,
5   firstname varchar (35),
6   surname varchar (30) not null,
7   gender varchar (6) not null,
8   age int not null,
9   salary decimal (8,3) not null,
10  primary key (person_id)
11 );
12 explain person;
```

The 'Result Grid' shows the output of the 'explain' statement:

Field	Type	Null	Key	Default	Extra
person_id	int	NO	PRI		
firstname	varchar(35)	YES			
surname	varchar(30)	NO			
gender	varchar(6)	NO			
age	int	NO			
salary	decimal(8,3)	NO			

The bottom status bar indicates 'Query Completed'.

Creating 'Public Company Stocks' Database

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' list with 'amazon1' selected. The main query editor contains the following SQL code:

```
1 CREATE DATABASE public_company_stocks;
2 SHOW DATABASES;
```

The 'Result Grid' shows the output of the 'SHOW DATABASES' command:

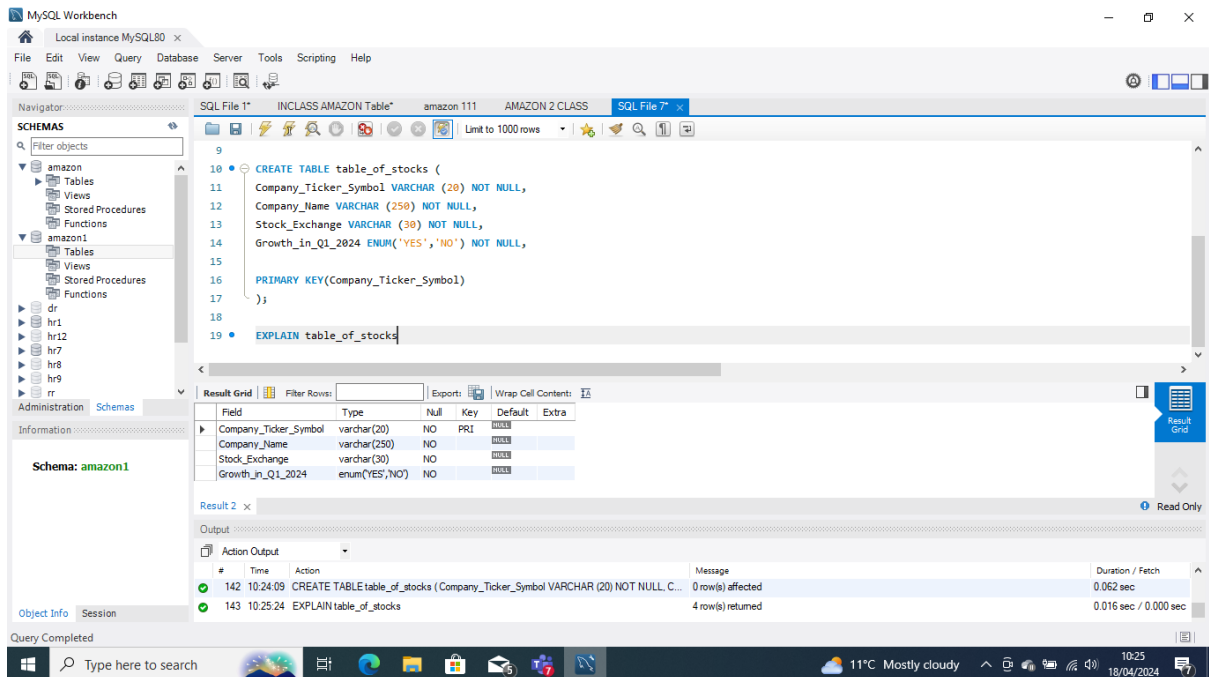
Database
public_company_st...
rr
sakila
sys
world

The 'Output' pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
135	10:03:54	CREATE DATABASE public_company_stocks	1 row(s) affected	0.063 sec
136	10:03:54	SHOW DATABASES	17 row(s) returned	0.015 sec / 0.015 sec

The bottom status bar indicates 'Query Completed'.

Creating a Table: Company Ticker Symbol, Company Name, Stock Exchange, Growth. Also Company Ticker Symbol is a Primary Key

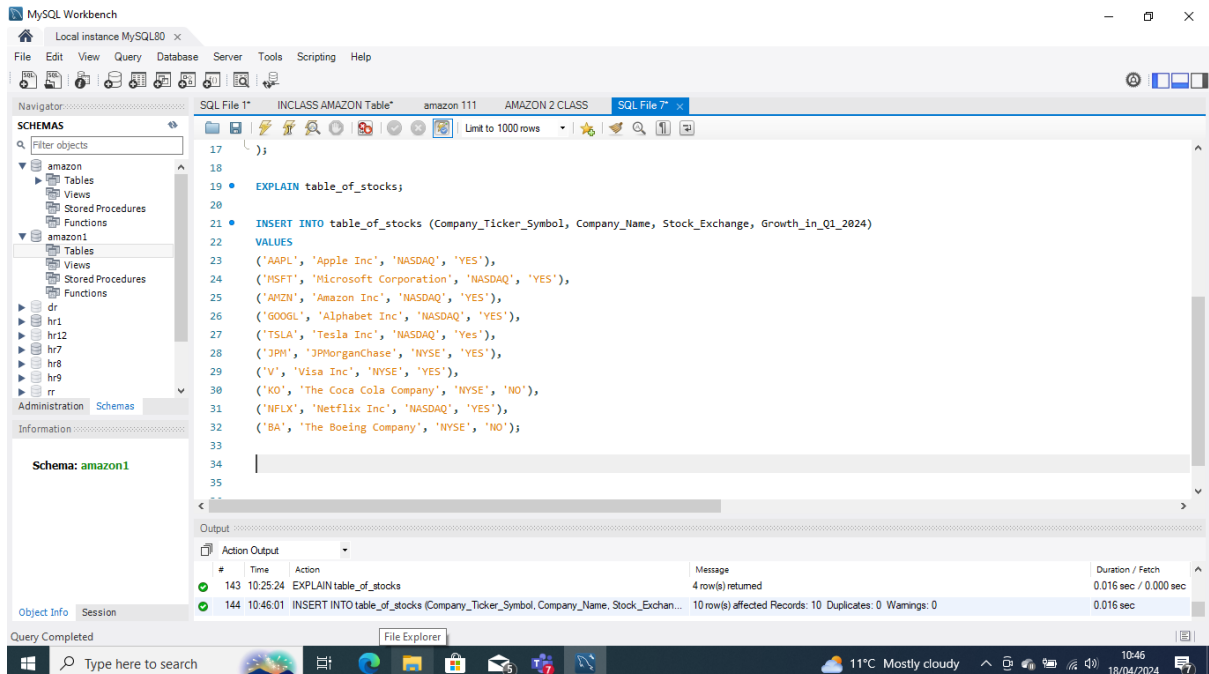


MySQL Workbench interface showing the creation of a table named 'table_of_stocks' in the 'amazon1' schema. The table structure is defined as follows:

Field	Type	Null	Key	Default	Extra
Company_Ticker_Symbol	varchar(20)	NO	PRI		
Company_Name	varchar(250)	NO			
Stock_Exchange	varchar(30)	NO			
Growth_in_Q1_2024	enum('YES','NO')	NO			

The output shows the execution of the CREATE TABLE statement and the EXPLAIN statement, indicating that 4 rows were returned.

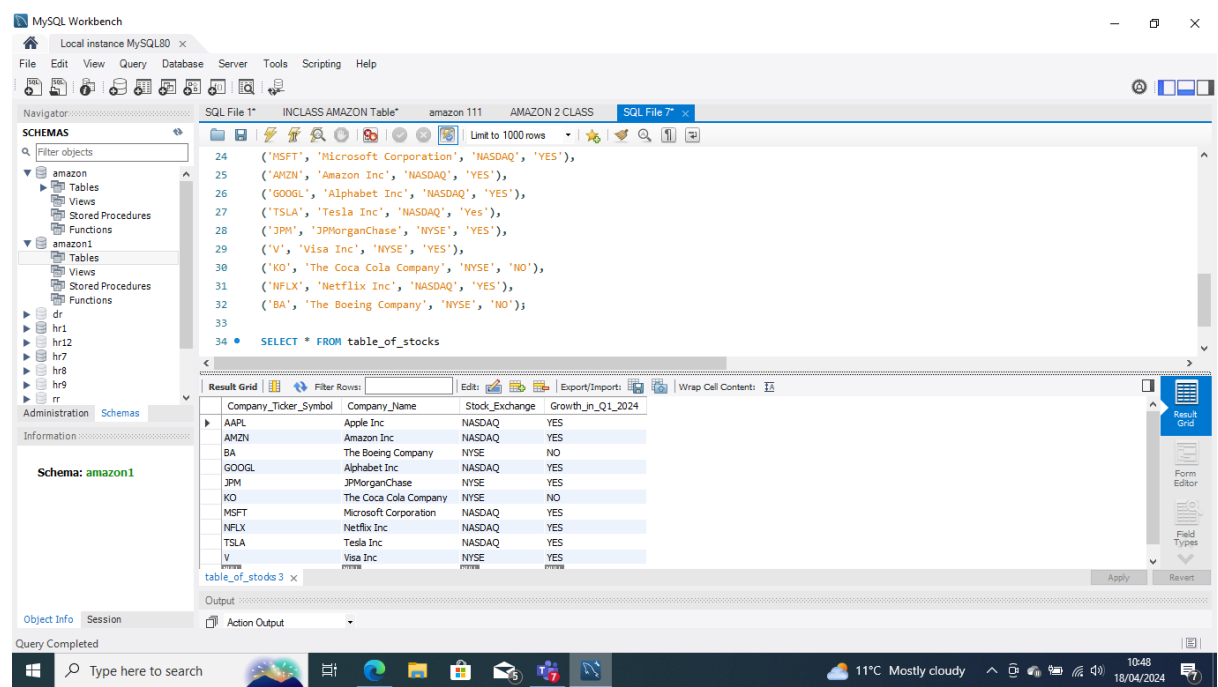
Inserting the data in to table, using 'insert into'



MySQL Workbench interface showing the insertion of data into the 'table_of_stocks' table. The INSERT INTO statement is executed, and the output shows 10 rows affected. The data includes company ticker symbols, names, stock exchanges, and growth in Q1 2024.

#	Time	Action	Message	Duration / Fetch
143	10:25:24	EXPLAIN table_of_stocks	4 row(s) returned	0.016 sec / 0.000 sec
144	10:46:01	INSERT INTO table_of_stocks (Company_Ticker_Symbol, Company_Name, Stock_Exchange, Growth_in_Q1_2024) VALUES ('AAPL', 'Apple Inc', 'NASDAQ', 'YES'), ('MSFT', 'Microsoft Corporation', 'NASDAQ', 'YES'), ('AMZN', 'Amazon Inc', 'NASDAQ', 'YES'), ('GOOGL', 'Alphabet Inc', 'NASDAQ', 'YES'), ('TSLA', 'Tesla Inc', 'NASDAQ', 'YES'), ('JPM', 'JPMorgan Chase', 'NYSE', 'YES'), ('V', 'Visa Inc', 'NYSE', 'YES'), ('KO', 'The Coca Cola Company', 'NYSE', 'NO'), ('NFLX', 'Netflix Inc', 'NASDAQ', 'YES'), ('BA', 'The Boeing Company', 'NYSE', 'NO');	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.016 sec

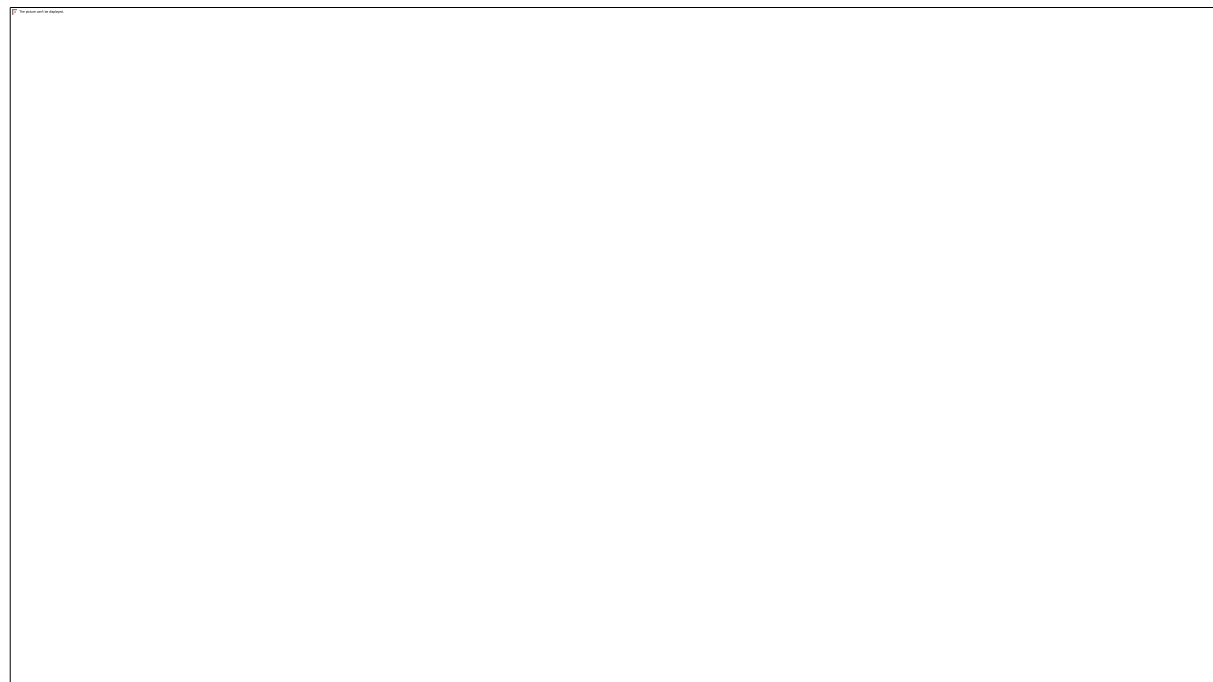
Showing all the Data using the Select*From



Change or update the data in the table



Drop the column Company Name



Add Column

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: SQL File 1* INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES* HUMAN REDSUCES

SCHEMAS

Filter objects

amazon

- Tables
- Views
- Stored Procedures
- Functions

amazon1

- Tables
- Views
- Stored Procedures
- Functions

dr

- hr1
- hr12
- hr7
- hr8
- hr9
- rr

Administration Schemas

Information

Schema: amazon1

Object Info Session

Query Completed

SQL File 1* INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES* HUMAN REDSUCES

```
55 ALTER TABLE table_of_stocks
56 ADD COLUMN Company_Value decimal(7,2);
57
58 EXPLAIN table_of_stocks;
59
60
61
62
63
```

Result Grid

Field	Type	Null	Key	Default	Extra
Company_Ticker_Symbol	varchar(20)	NO	PRI		
Stock_Exchange	varchar(30)	NO			
Growth_in_Q1_2024	enum('YES','NO')	NO			
Company_Value	decimal(10,0)	YES			

Result 6 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
183	14:10:29	ALTER TABLE table_of_stocks ADD COLUMN Company_Value decimal(7,2)	Error Code: 1060. Duplicate column name 'Company_Value'	0.000 sec
184	14:14:51	EXPLAIN table_of_stocks	4 row(s) returned	0.000 sec / 0.000 sec

11°C. Mostly cloudy

14:15

18/04/2024

Add data in to the new column

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```
105 SET Company_Value = '10.00';
106 WHERE Company_Ticker_Symbol = 'TSLA';
107
108
109 • UPDATE table_of_stocks
110 SET Company_Value = '10.00'
111 WHERE Company_Ticker_Symbol = 'TSLA';
112
113
114 • SELECT * FROM table_of_stocks;
```

The Result Grid shows the following data:

Company_Ticker_Symbol	Stock_Exchange	Growth_in_Q1_2024	Company_Value
AAPL	NASDAQ	YES	3
AMZN	NASDAQ	YES	5
BA	NYSE	NO	50
GOOGL	NASDAQ	YES	9
JPM	NYSE	YES	1
KO	NYSE	NO	5
MSFT	NASDAQ	YES	7
NFLX	NASDAQ	YES	10

DELETE KO from stock list

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```
115
116 • DELETE FROM table_of_stocks
117 WHERE Company_Ticker_Symbol = 'KO';
118 • SELECT * FROM table_of_stocks;
```

The Result Grid shows the following data:

Company_Ticker_Symbol	Stock_Exchange	Growth_in_Q1_2024	Company_Value
AAPL	NASDAQ	YES	3
AMZN	NASDAQ	YES	5
BA	NYSE	NO	50
GOOGL	NASDAQ	YES	9
JPM	NYSE	YES	1
MSFT	NASDAQ	YES	7
NFLX	NASDAQ	YES	10
TSLA	NASDAQ	NO	10
V	NYSE	YES	8
NULL	NULL	NULL	NULL

select stocks that have value >7

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

amazon
Tables
Views
Stored Procedures
Functions
amazon1
Tables
Views
Stored Procedures
Functions
dr
hr1
hr12
hr7
hr8
hr9
rr

Administration Schemas

Information

Schema: amazon1

SQL File 1* INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES* HUMAN REOSUCES

120
121 • SELECT Company_Ticker_Symbol, Company_Value
122 FROM table_of_stocks
123 WHERE Company_Value > 7;
124
125
126
127
128
129

Result Grid Filter Rows: Edit Export/Import: Wrap Cell Content: 12

Company_Ticker_Symbol	Company_Value
BA	50
GOOGL	9
NFLX	10
TSLA	10
V	8
NULL	NULL

table_of_stocks 17 x

Output

Query Completed

Type here to search

11°C Mostly cloudy 14:37 18/04/2024

SELECT ALL nasdaq STOCK

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

amazon
Tables
Views
Stored Procedures
Functions
amazon1
Tables
Views
Stored Procedures
Functions
dr
hr1
hr12
hr7
hr8
hr9
rr

Administration Schemas

Information

Schema: amazon1

SQL File 1* INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES* HUMAN REOSUCES

123 WHERE Company_Value > 7;
124
125 • SELECT *
126 FROM table_of_stocks
127 WHERE Stock_Exchange = 'NASDAQ';
128
129
130
131

Result Grid Filter Rows: Edit Export/Import: Wrap Cell Content: 12

Company_Ticker_Symbol	Stock_Exchange	Growth_in_Q1_2024	Company_Value
AAPL	NASDAQ	YES	3
AMZN	NASDAQ	YES	5
GOOGL	NASDAQ	YES	9
MSFT	NASDAQ	YES	7
NFLX	NASDAQ	YES	10
TSLA	NASDAQ	NO	10
NULL	NULL	NULL	NULL

table_of_stocks 18 x

Output

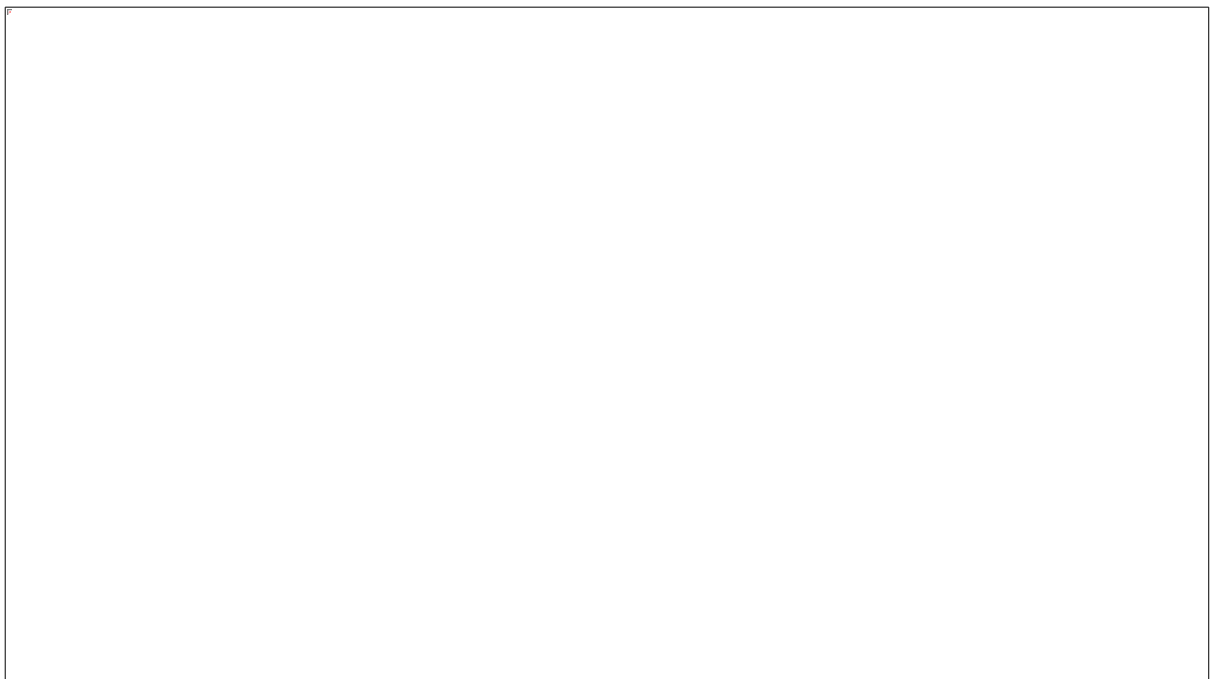
Action Output

Query Completed

Type here to search

11°C Mostly cloudy 14:40 18/04/2024

New Database

A large, empty rectangular box with a thin black border, occupying the lower half of the page. It is intended for a user to provide details for a new database, such as name, location, and other configuration parameters.

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES SQL File 8*

SCHEMAS

Filter objects

amazon

Tables

Views

Stored Procedures

Functions

amazon1

Tables

Views

Stored Procedures

Functions

dr

hr1

hr12

hr7

hr8

hr9

rr

Administration Schemas

Information

Schema: amazon1

SQL File 1*

```
3 • USE human_resources;
4 • CREATE TABLE Human_Resources_Table (
5   ID INT NOT NULL,
6   firstname VARCHAR (100) NOT NULL,
7   surname VARCHAR (100) NOT NULL,
8   gender VARCHAR (10) NOT NULL,
9   age INT NOT NULL,
10  salary DECIMAL (8,3) NOT NULL,
11  PRIMARY KEY (ID)
12 );
13
14 • EXPLAIN Human_Resources_Table;
```

Result Grid

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
firstname	varchar(100)	NO		NULL	
surname	varchar(100)	NO		NULL	
gender	varchar(10)	NO		NULL	
age	int	NO		NULL	

Result 2 x

Output

Action Output

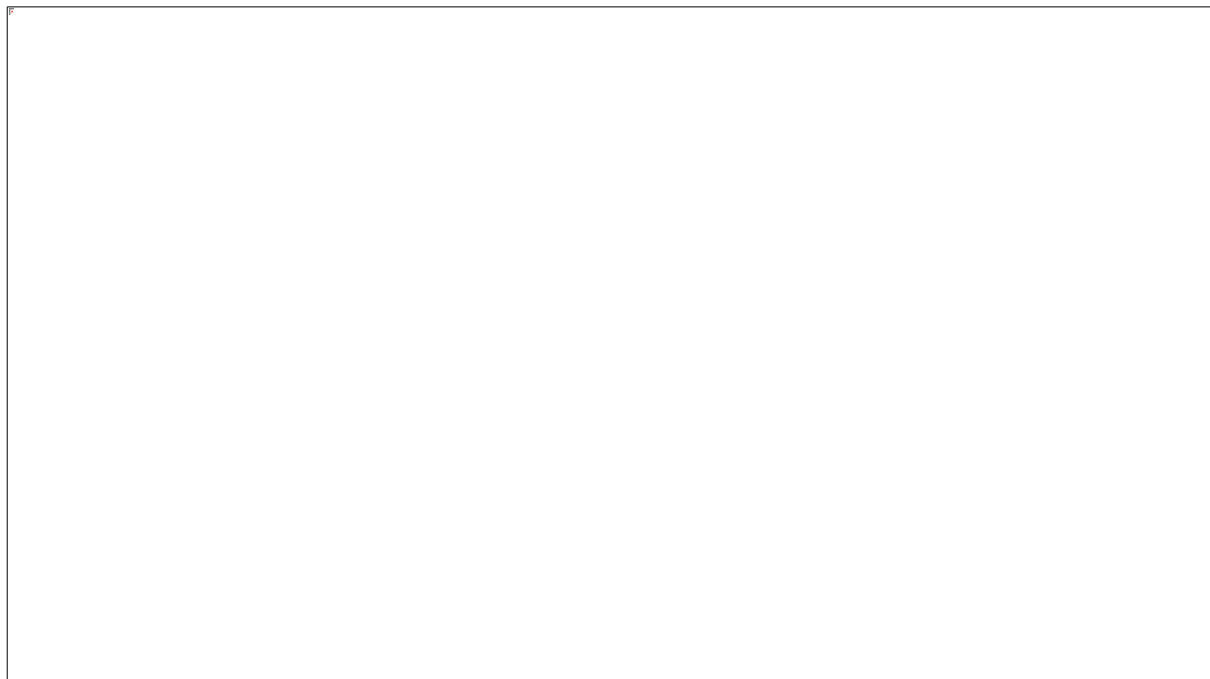
#	Time	Action	Message	Duration / Fetch
159	13:23:35	CREATE TABLE Human_Resources_Table (ID INT NOT NULL,firstname VARCHAR (100)...	0 row(s) affected	0.046 sec
160	13:24:09	EXPLAIN Human_Resources_Table	6 row(s) returned	0.000 sec / 0.000 sec

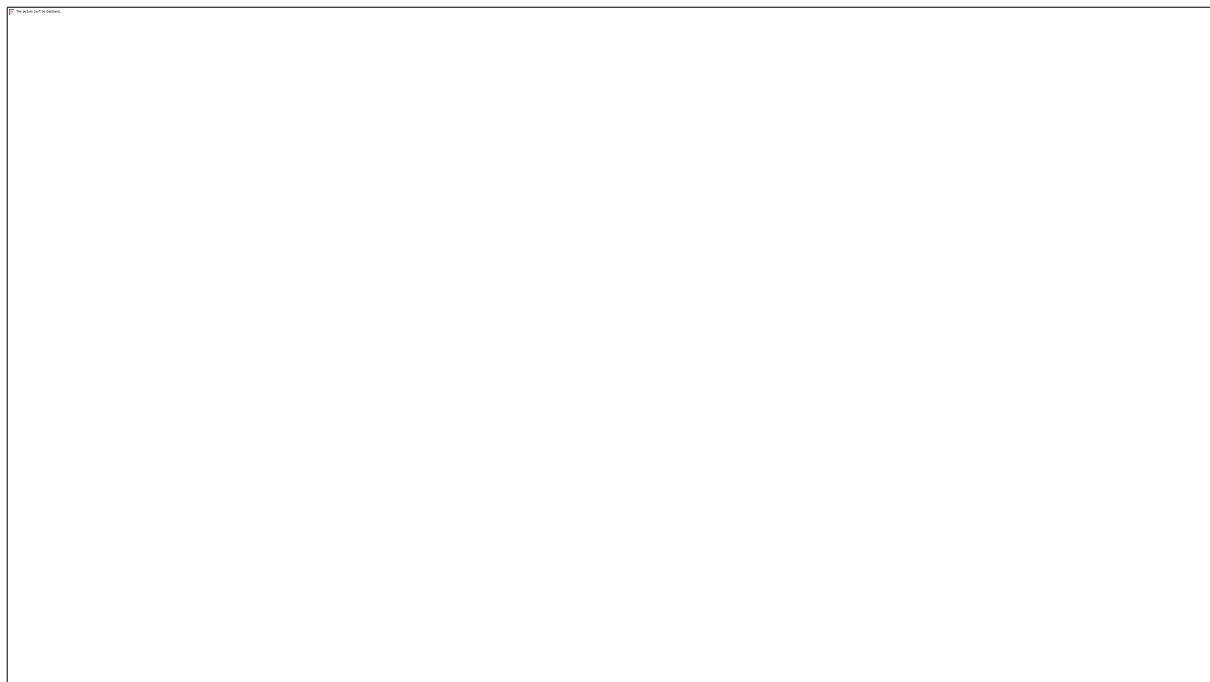
Query Completed

Object Info Session

FTSE 100 -1.56%

13:24 18/04/2024





MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

amazon

Tables

Views

Stored Procedures

Functions

amazon1

Tables

Views

Stored Procedures

Functions

dr

hr1

hr12

hr7

hr8

hr9

rr

Administration Schemas

Information

Schema: amazon1

Object Info Session

Query Completed

SQL File 1* INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES SQL File 8* x

Limit to 1000 rows

```
30
31 • SELECT firstname, surname, salary
32 FROM Human_Resources_Table
33 WHERE salary >60000.00;
34
35 • SELECT *
36 FROM Human_Resources_Table
37 WHERE age >50 AND age< 70;
38
39
```

Result Grid

ID	firstname	surname	gender	age	salary
1	Borris	Johnson	Male	60	30000.000
3	Serena	Williams	Female	55	20000.000
4	Will	Smith	Male	56	70000.000
7	William	Maher	Male	67	50000.000
8	Joe	Rogan	Male	55	50000.000

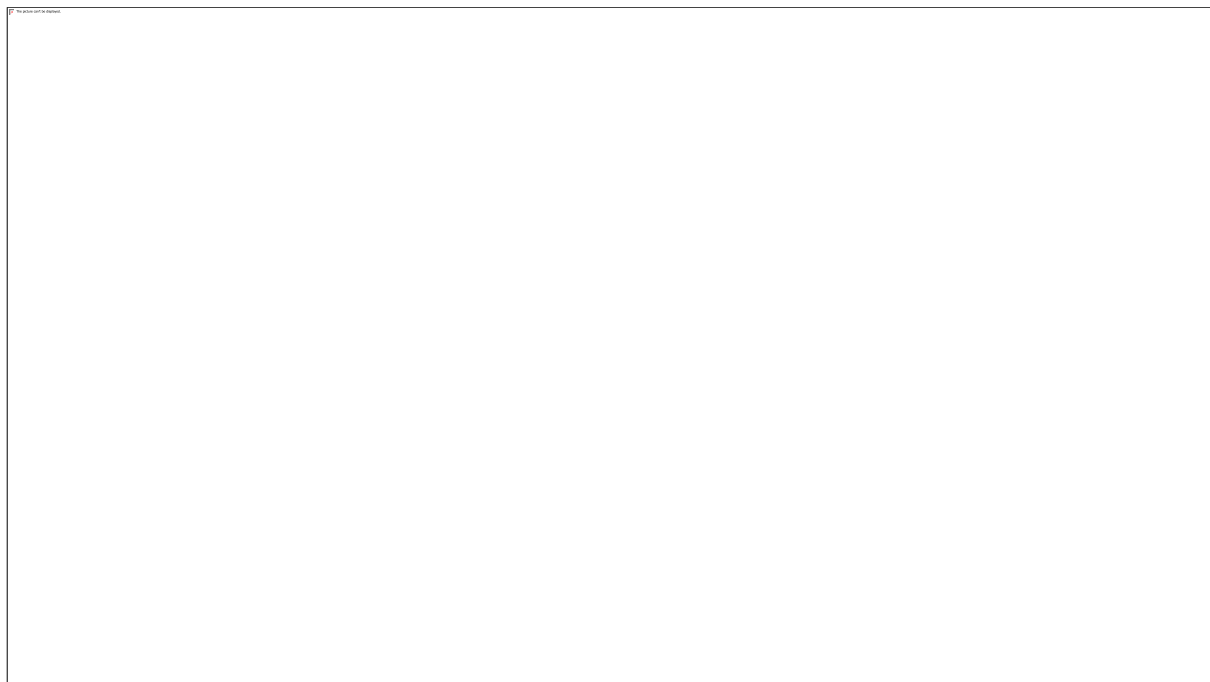
Human_Resources_Table 11 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
172	13:47:46	SELECT firstname, surname, salary FROM Human_Resources_Table WHERE salary >60.0...	10 row(s) returned	0.000 sec / 0.000 sec
173	13:47:49	SELECT firstname, surname, salary FROM Human_Resources_Table WHERE salary >60.0...	10 row(s) returned	0.000 sec / 0.000 sec
174	13:48:23	SELECT firstname, surname, salary FROM Human_Resources_Table WHERE salary >600...	4 row(s) returned	0.000 sec / 0.000 sec
175	13:51:42	SELECT * FROM Human_Resources_Table WHERE age >50 AND age< 70 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

11°C Mostly cloudy 13:51 18/04/2024



sum of salary

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: SQL File 1* INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES HUMAN REOSUCES*

SCHEMAS

Filter objects

- amazon
 - Tables
 - Views
 - Stored Procedures
 - Functions
- amazon1
 - Tables
 - Views
 - Stored Procedures
 - Functions
- dr
- hr1
- hr12
- hr7
- hr8
- hr9
- rr

Administration Schemas

Information

Schema: amazon1

Result Grid

Filter Rows: Exports: Wrap Cell Contents

SUM(salary)

549999.990

Result 20 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
234	15:46:11	SELECT * FROM Human_Resources_Table WHERE age = 55 OR AGE = 79 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
235	16:45:01	SELECT SUM(salary) From Human_Resources_Table LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Query Completed

11°C Mostly cloudy 16:45 18/04/2024

select min salary

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80 x

Filter objects

amazon

- Tables
- Views
- Stored Procedures
- Functions

amazon1

- Tables
- Views
- Stored Procedures
- Functions

dr

- hr1
- hr12
- hr7
- hr8
- hr9
- rr

Administration Schemas

Information

Schema: amazon1

SQL File 1*

INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES HUMAN REOSUCES*

51 FROM Human_Resources_Table

52 WHERE age = '55' OR AGE = '79';

53

54 • SELECT SUM(salary)

55 From Human_Resources_Table;

56

57 • SELECT min(salary)

58 From Human_Resources_Table;

59

60

61

Result Grid

min(salary)

20000.000

Result 21 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
236	16:46:26	SELECT MIN(salary) From Human_Resources LIMIT 0, 1000	Error Code: 1146. Table 'human_resources.human_resources' doesn't exist	0.015 sec
237	16:46:54	SELECT min(salary) From Human_Resources_Table LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

Object Info Session

Query Completed

Type here to search

11°C Mostly cloudy

16:47 18/04/2024

max salary

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80 x

Filter objects

amazon

- Tables
- Views
- Stored Procedures
- Functions

amazon1

- Tables
- Views
- Stored Procedures
- Functions

dr

- hr1
- hr12
- hr7
- hr8
- hr9
- rr

Administration Schemas

Information

Schema: amazon1

SQL File 1*

INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES HUMAN REOSUCES*

54 • SELECT SUM(salary)

55 From Human_Resources_Table;

56

57 • SELECT min(salary)

58 From Human_Resources_Table;

59

60 • SELECT max(salary)

61 From Human_Resources_Table;

62

63

64

Result Grid

max(salary)

99999.990

Result 22 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
237	16:46:54	SELECT min(salary) From Human_Resources_Table LIMIT 0, 1000	1 row(s) returned	
238	16:47:49	SELECT max(salary) From Human_Resources_Table LIMIT 0, 1000	1 row(s) returned	

Object Info Session

Query Completed

Type here to search

11°C Mostly cloudy

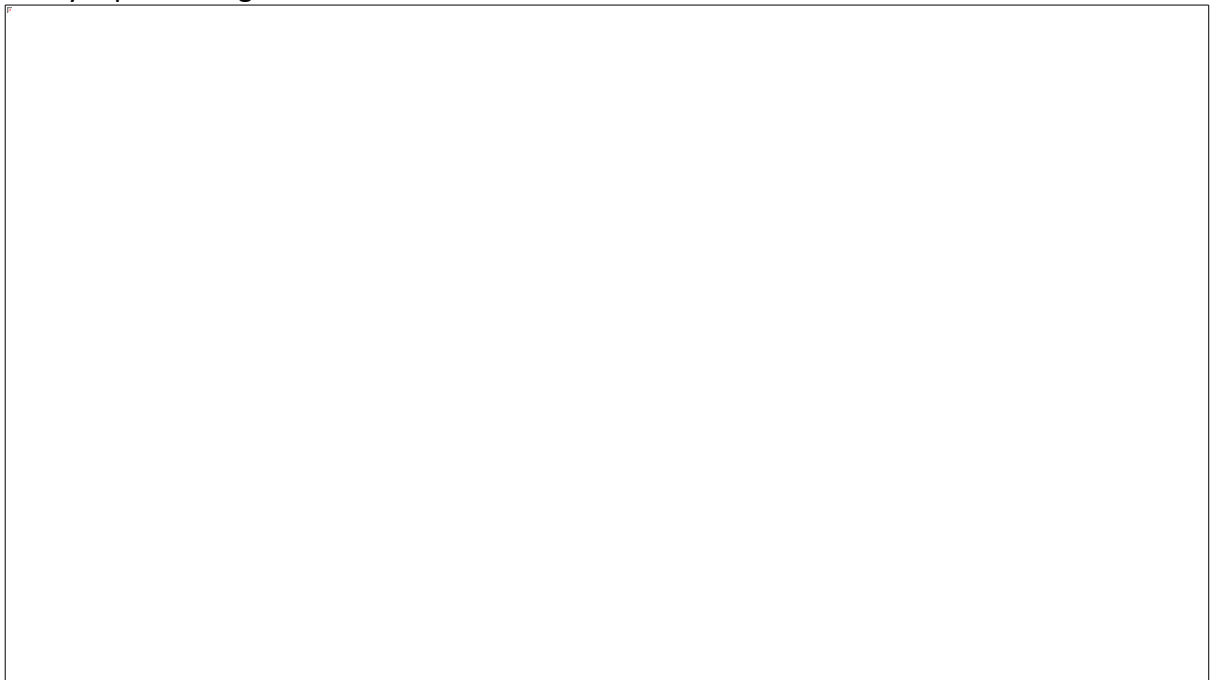
16:47 18/04/2024

Shahid B replied to a conversation you're in CID - 25/03/2024 - AFT - Johan A / General

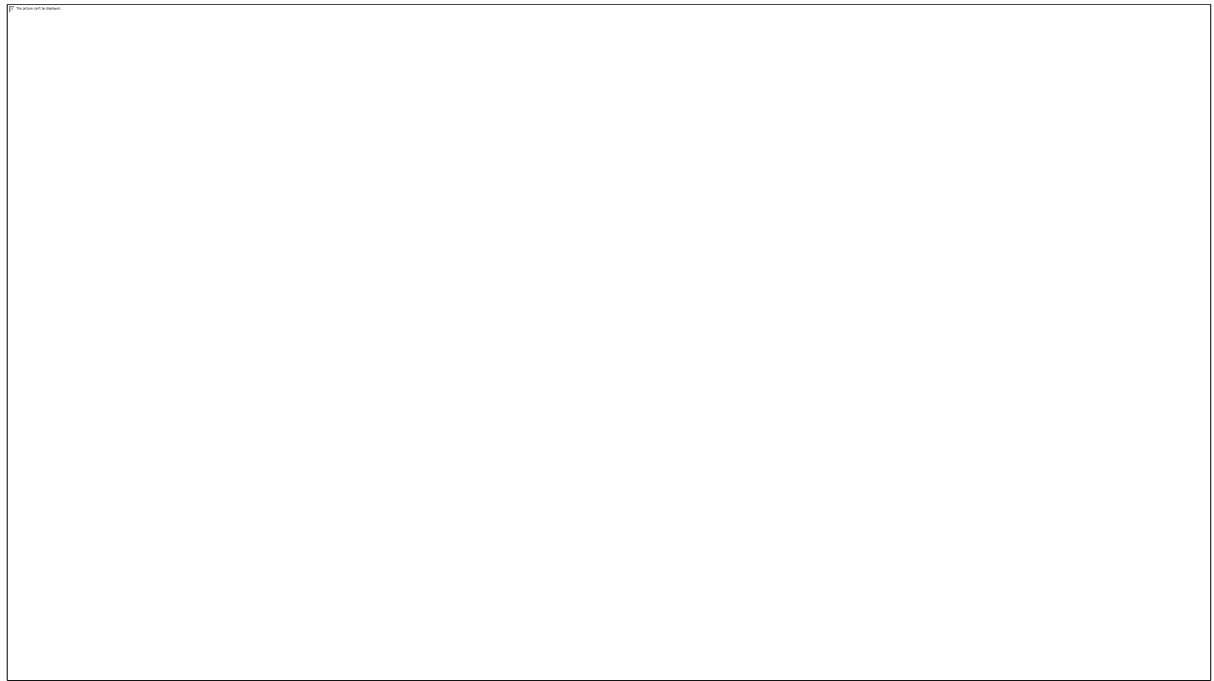
number of salaries



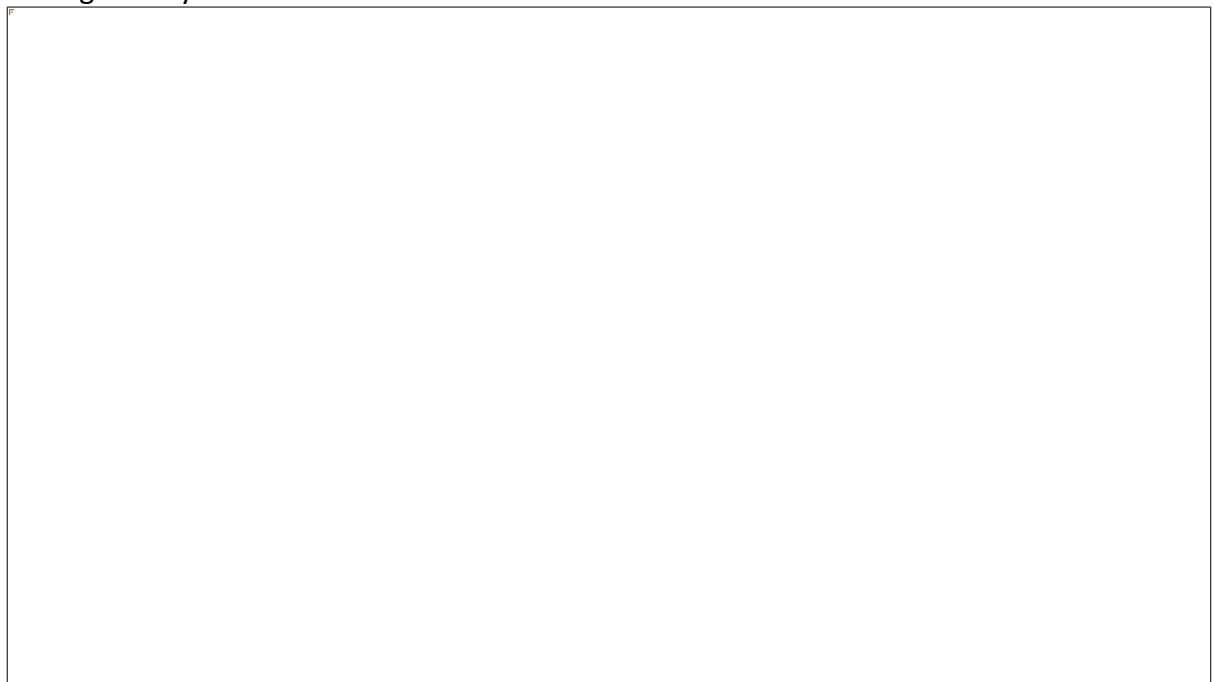
salary equal to or greater than 50k



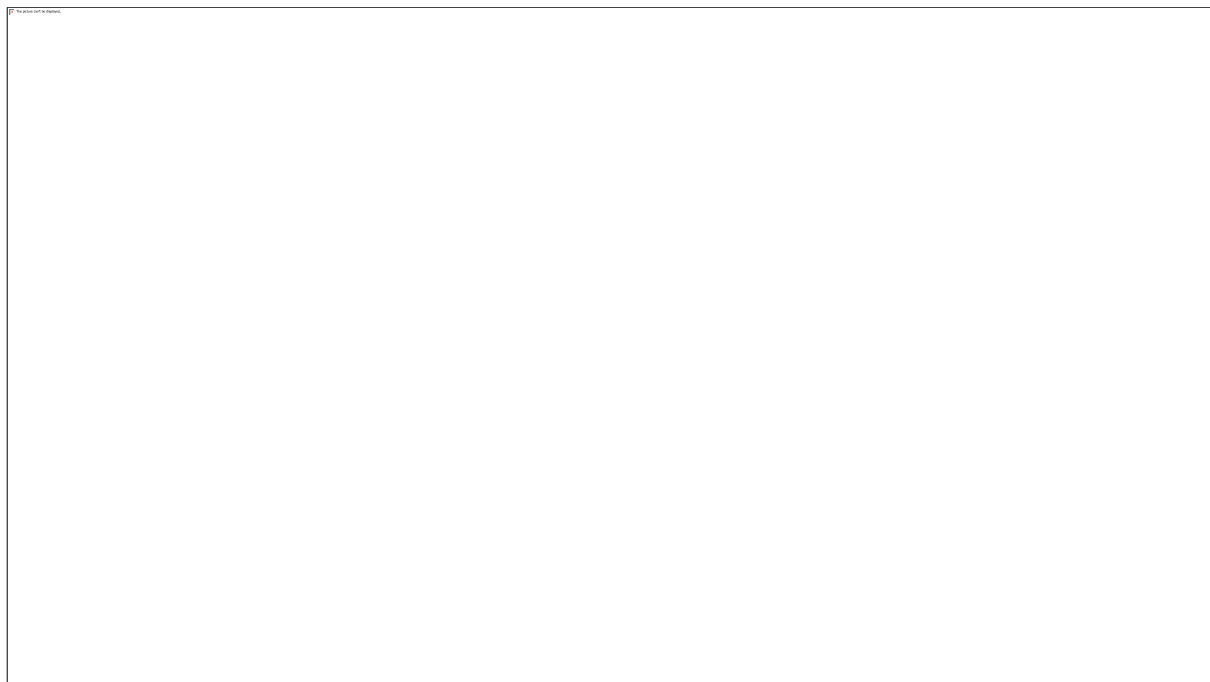
where salary is greater than 90k



average salary



University Example



MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- amazon
- amazon1
- amazon2
- dr
- hr1
- hr12
- hr7
- hr8
- hr9
- human_resources
- public_company_stodos
- rr
- sakila
- sys
- university
- Tables
- Views

Administration Schemas

Information

No object selected

AMAZON 2 CLASS* STOCK TABLES HUMAN REOSURES* uni JOINS TRAVELLER* GROUP Group Work*

Limit to 1000 rows

```
115 • SELECT * FROM students WHERE fname like "b%";
116
117 -- JOINS ----
118 -- join tables students and classes together--
119
120 • SELECT s.student_id, s.fname, class_id FROM enrolment
121 JOIN students s ON s.student_id=class_id
```

Result Grid

student_id	fname	class_id
1	Bradd	1
1	Bradd	2
1	Bradd	3
1	Bradd	4
2	George	2
2	George	4
3	Beyonce	1
3	Beyonce	2
4	Jeff	1
4	Jeff	2
4	Jeff	3

Result 23 x

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
135	13:19:29	SELECT * FROM enrolment LIMIT 0, 1000	20 row(s) returned	0.000 sec / 0.000 sec
136	13:20:03	SELECT * FROM students WHERE fname like "b%" LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
137	13:20:39	SELECT s.student_id, s.fname, class_id FROM enrolment JOIN students s ON s.student_id=...	20 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

13°C Windy 13:20 29/04/2024

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: AMAZON 2 CLASS* STOCK TABLES HUMAN REOSUCES* uni x JOINS TRAVELLER* GROUP Group Work*

SCHMAS

Filter objects

amazon
amazon1
amazon2
dr
hr1
hr12
hr7
hr8
hr9
human_resources
public_company_stods
rr
sakila
sys
university
Tables
Views
Administration Schemas

Information

No object selected

Result Grid

student_id fname class_id name

8	David	1	mathematics
8	David	2	physics
7	David	2	physics
7	David	3	electronics
6	Bill	1	mathematics
6	Bill	2	physics
6	Bill	3	electronics
6	Bill	4	computer_science
5	Mark	2	physics
4	Jeff	1	mathematics
4	Jeff	2	physics

Result 28 x

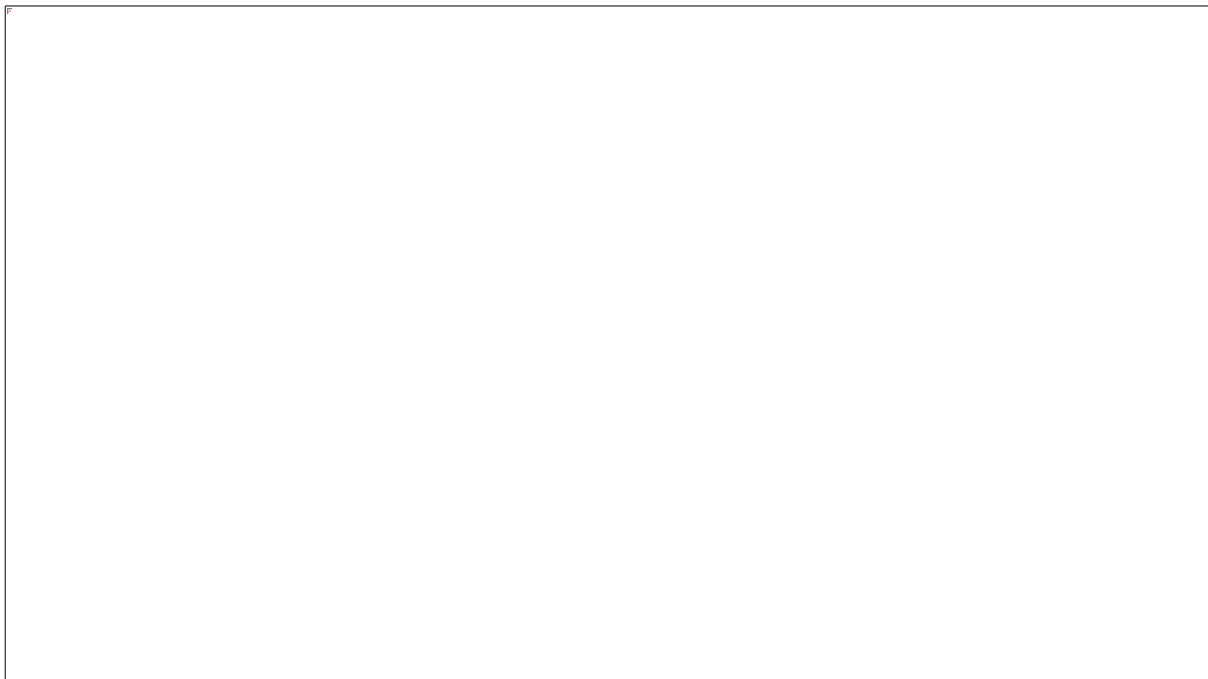
Output

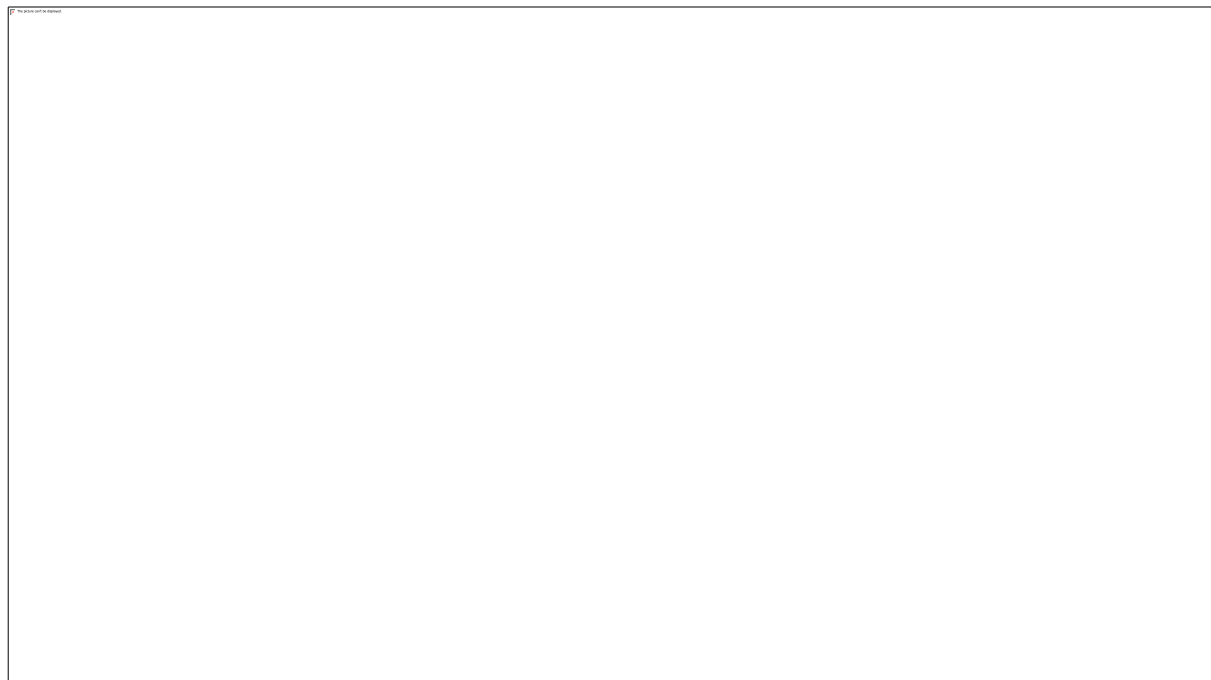
Action Output

#	Time	Action	Message	Duration / Fetch
140	13:21:54	SELECT s.student_id, s.fname, c.class_id, c.name FROM enrolment JOIN students s ON s.student_id=enrolment.student_id	20 row(s) returned	0.000 sec / 0.000 sec
141	13:22:21	SELECT s.student_id, s.fname, c.class_id, c.name FROM enrolment JOIN students s ON s.student_id=enrolment.student_id	20 row(s) returned	0.000 sec / 0.000 sec
142	13:22:41	SELECT s.student_id, s.fname, c.class_id, c.name FROM enrolment JOIN students s ON s.student_id=enrolment.student_id	20 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

13°C Windy 13:22 29/04/2024





sum of salary

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: INCLASS AMAZON Table* amazon 111 AMAZON 2 CLASS STOCK TABLES HUMAN REOSUCES*

SCHEMAS

Filter objects

amazon

Tables

Views

Stored Procedures

Functions

amazon1

Tables

Views

Stored Procedures

Functions

dr

hr1

hr12

hr7

hr8

hr9

rr

Administration Schemas

Schema: amazon1

Object Info Session

SQL File 1*

```
48 WHERE age >= 55;
49
50 • SELECT *
51 FROM Human_Resources_Table
52 WHERE age = '55' OR AGE = '79';
53
54 • SELECT SUM(salary)
55 From Human_Resources_Table;
56
57
58
```

Result Grid

Filter Rows: Exports Wrap Cell Contents

	SUM(salary)
▶	549999.990

Result 20 x Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
234	15:46:11	SELECT * FROM Human_Resources_Table WHERE age = '55' OR AGE = '79' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
235	16:45:01	SELECT SUM(salary) From Human_Resources_Table LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Query Completed

11°C Mostly cloudy 16:45 18/04/2024

select min salary

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
51 FROM Human_Resources_Table
52 WHERE age = '55' OR AGE = '79';
53
54 • SELECT SUM(salary)
55 From Human_Resources_Table;
56
57 • SELECT min(salary)
58 From Human_Resources_Table;
59
60
61
```

The Results tab shows the output of the query:

min(salary)
20000.000

The Action Output tab shows the execution details:

#	Time	Action	Message	Duration / Fetch
236	16:46:26	SELECT MIN(salary) From Human_Resources LIMIT 0, 1000	Error Code: 1146. Table 'human_resources.human_resources' doesn't exist	0.015 sec
237	16:46:54	SELECT min(salary) From Human_Resources_Table LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

max salary

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
54 • SELECT SUM(salary)
55 From Human_Resources_Table;
56
57 • SELECT min(salary)
58 From Human_Resources_Table;
59
60 • SELECT max(salary)
61 From Human_Resources_Table;
62
63
64
```

The Results tab shows the output of the query:

max(salary)
99999.990

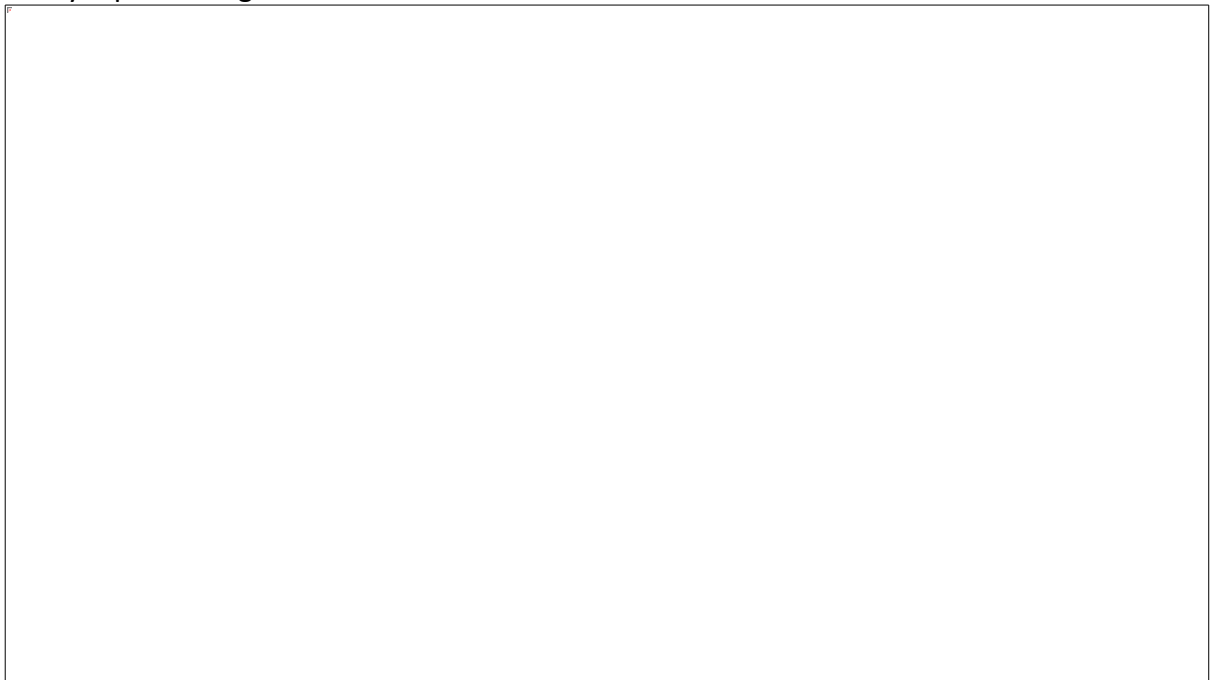
The Action Output tab shows the execution details:

#	Time	Action	Message	Duration / Fetch
237	16:46:54	SELECT min(salary) From Human_Resources_Table LIMIT 0, 1000	1 row(s) returned	
238	16:47:49	SELECT max(salary) From Human_Resources_Table LIMIT 0, 1000	1 row(s) returned	

number of salaries



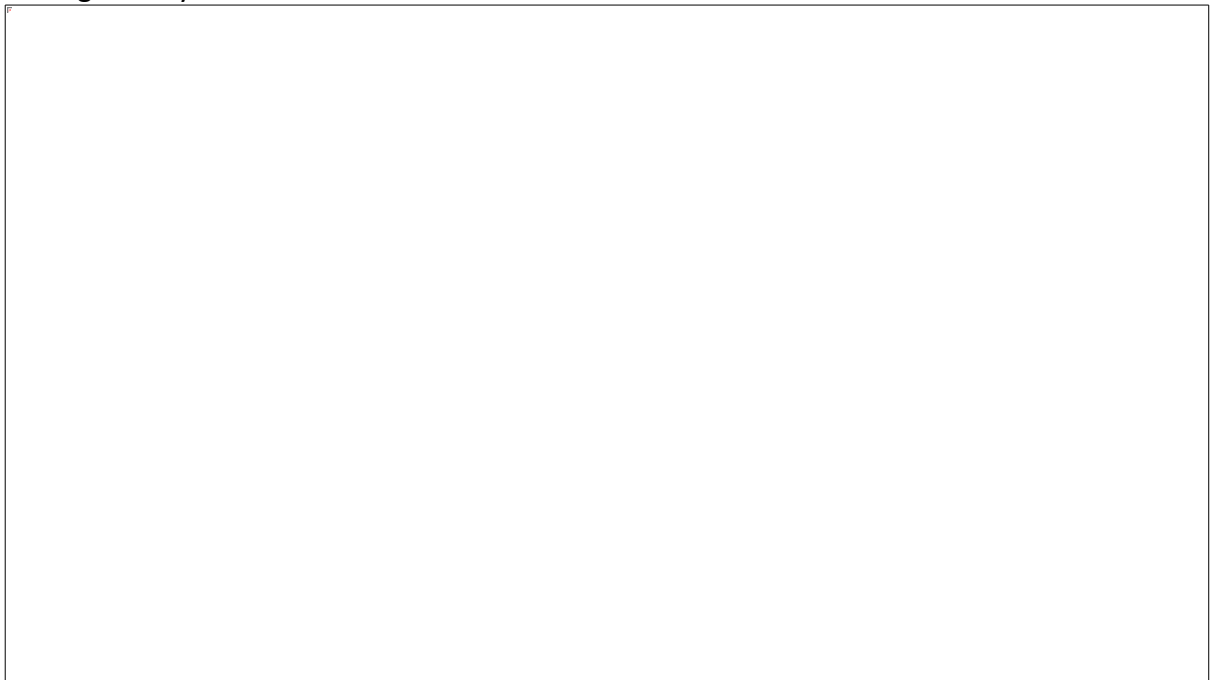
salary equal to or greater than 50k



where salary is greater than 90k



average salary



University Example

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: AMAZON 2 CLASS* STOCK TABLES HUMAN REOSUCES* uni JOINS TRAVELLER* GROUP Group Work*

SCHEMAS

Filter objects

- amazon
- amazon1
- amazon2
- dr
- hr1
- hr12
- hr7
- hr8
- hr9
- human_resources
- public_company_stods
- rr
- sakila
- sys
- university
- Tables
- Views

Information

No object selected

Result Grid

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	auto_increment	
fname	varchar(30)	NO			
lname	varchar(30)	NO			
dob	date	NO			

Result 16 x

Output

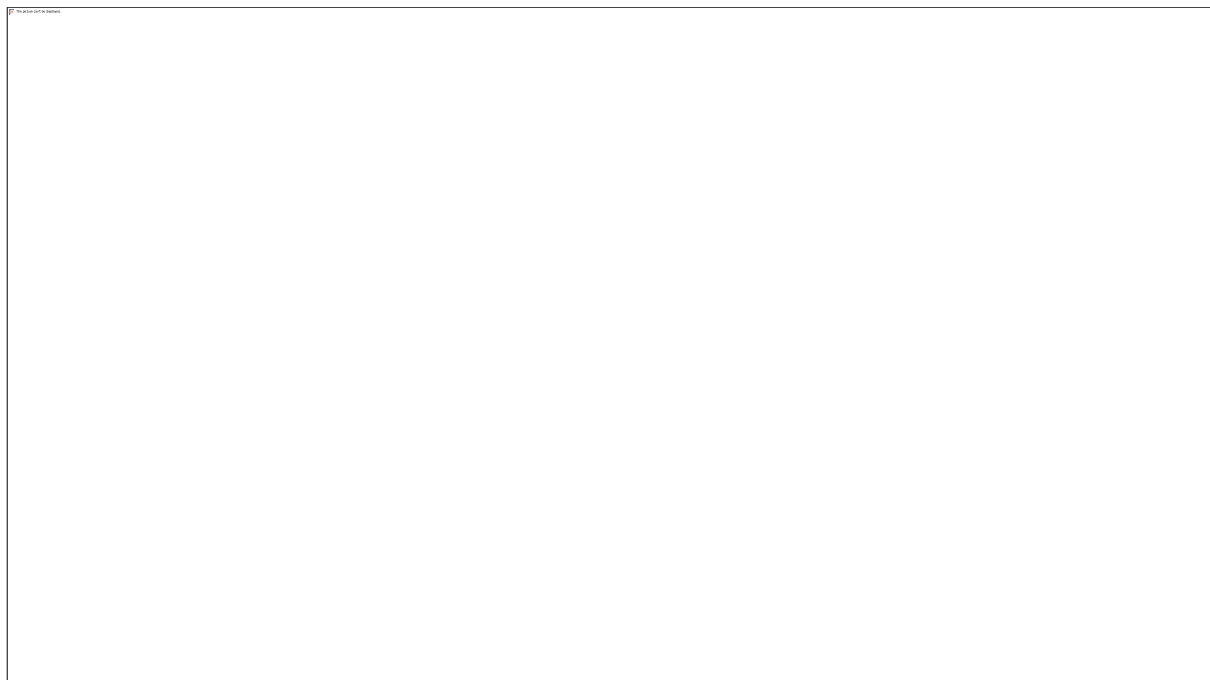
Action Output

#	Time	Action	Message	Duration / Fetch
120	13:05:45	SELECT * FROM orders WHERE magazine_code = 2 OR magazine_code = 3 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
121	13:06:17	SELECT * FROM magazines LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

13°C Windy 13:16 29/04/2024





MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- amazon
- amazon1
- amazon2
- dr
- hr1
- hr12
- hr7
- hr8
- hr9
- human_resources
- public_company_stodos
- rr
- sakila
- sys
- university
- Tables
- Views

Administration Schemas

Information

No object selected

AMAZON 2 CLASS* STOCK TABLES HUMAN REOSUCES* uni JOINS TRAVELLER* GROUP Group Work*

Limit to 1000 rows

```
115 • SELECT * FROM students WHERE fname like "b%";
116
117 -- JOINS ----
118 -- join tables students and classes together--
119
120 • SELECT s.student_id, s.fname, class_id FROM enrolment
121 JOIN students ON s.student_id=students.student_id
```

Result Grid

	student_id	fname	class_id
1	Bradd	1	
1	Bradd	2	
1	Bradd	3	
1	Bradd	4	
2	George	2	
2	George	4	
3	Beyonce	1	
3	Beyonce	2	
4	Jeff	1	
4	Jeff	2	
4	Jeff	3	

Result 23 x

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
135	13:19:29	SELECT * FROM enrolment LIMIT 0, 1000	20 row(s) returned	0.000 sec / 0.000 sec
136	13:20:03	SELECT * FROM students WHERE fname like "b%" LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
137	13:20:39	SELECT s.student_id, s.fname, class_id FROM enrolment JOIN students s ON s.student_id=...	20 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

13°C Windy 13:20 29/04/2024

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: AMAZON 2 CLASS* STOCK TABLES HUMAN REOSUCES* uni x JOINS TRAVELLER* GROUP Group Work*

SCHEMAS

Filter objects

- amazon
- amazon1
- amazon2
- dr
- hr1
- hr12
- hr7
- hr8
- hr9
- human_resources
- public_company_stods
- rr
- sakila
- sys
- university
- Tables
- Views

Administration Schemas

Information

No object selected

Result Grid

student_id fname class_id name

8	David	1	mathematics
8	David	2	physics
7	David	2	physics
7	David	3	electronics
6	Bill	1	mathematics
6	Bill	2	physics
6	Bill	3	electronics
6	Bill	4	computer_science
5	Mark	2	physics
4	Jeff	1	mathematics
4	Jeff	2	physics

Result 28 x

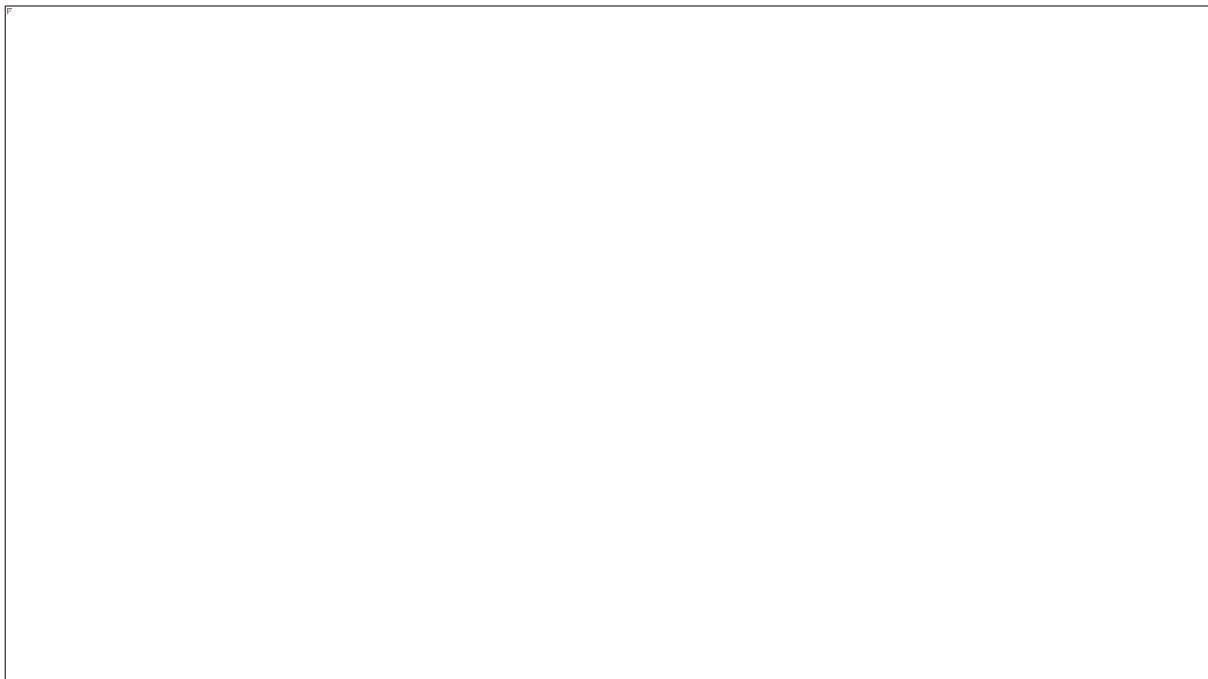
Output

Action Output

#	Time	Action	Message	Duration / Fetch
140	13:21:54	SELECT s.student_id, s.fname, c.class_id, c.name FROM enrolment JOIN students s ON s.student_id=enrolment.student_id	20 row(s) returned	0.000 sec / 0.000 sec
141	13:22:21	SELECT s.student_id, s.fname, c.class_id, c.name FROM enrolment JOIN students s ON s.student_id=enrolment.student_id	20 row(s) returned	0.000 sec / 0.000 sec
142	13:22:41	SELECT s.student_id, s.fname, c.class_id, c.name FROM enrolment JOIN students s ON s.student_id=enrolment.student_id	20 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

13°C Windy 13:22 29/04/2024



GROUP WORK

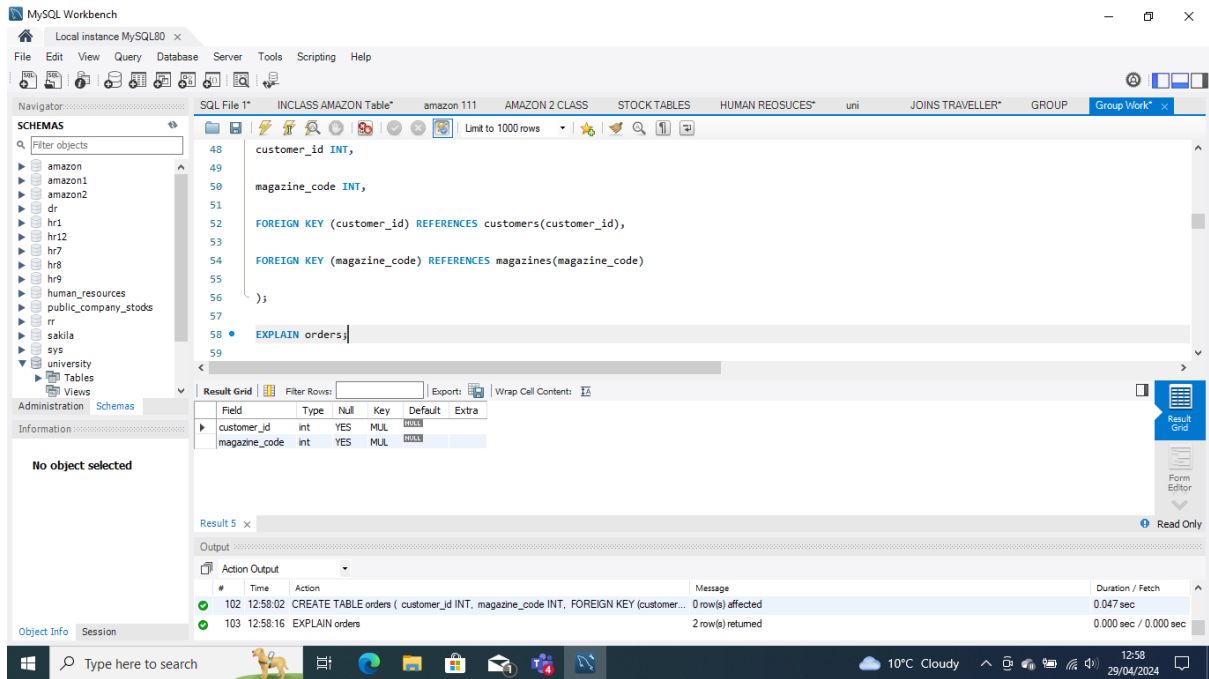


Created a table representing the customers

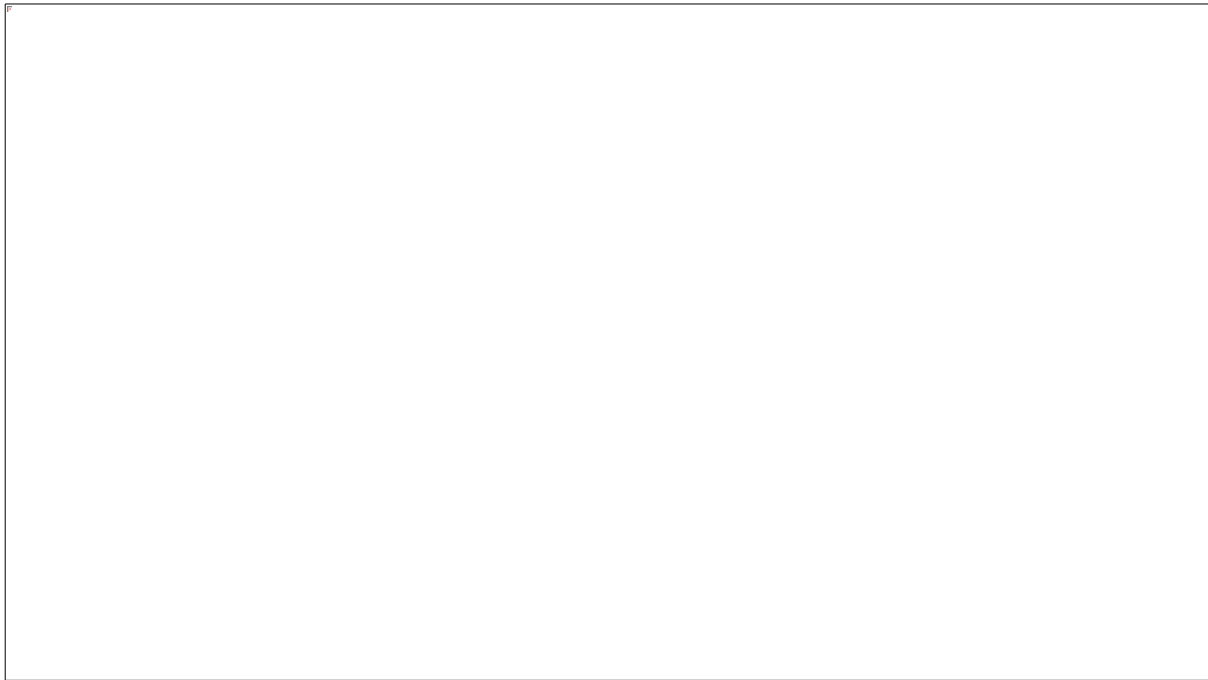
Created a table representing the magazine



made customer id and magazine id the foreign key

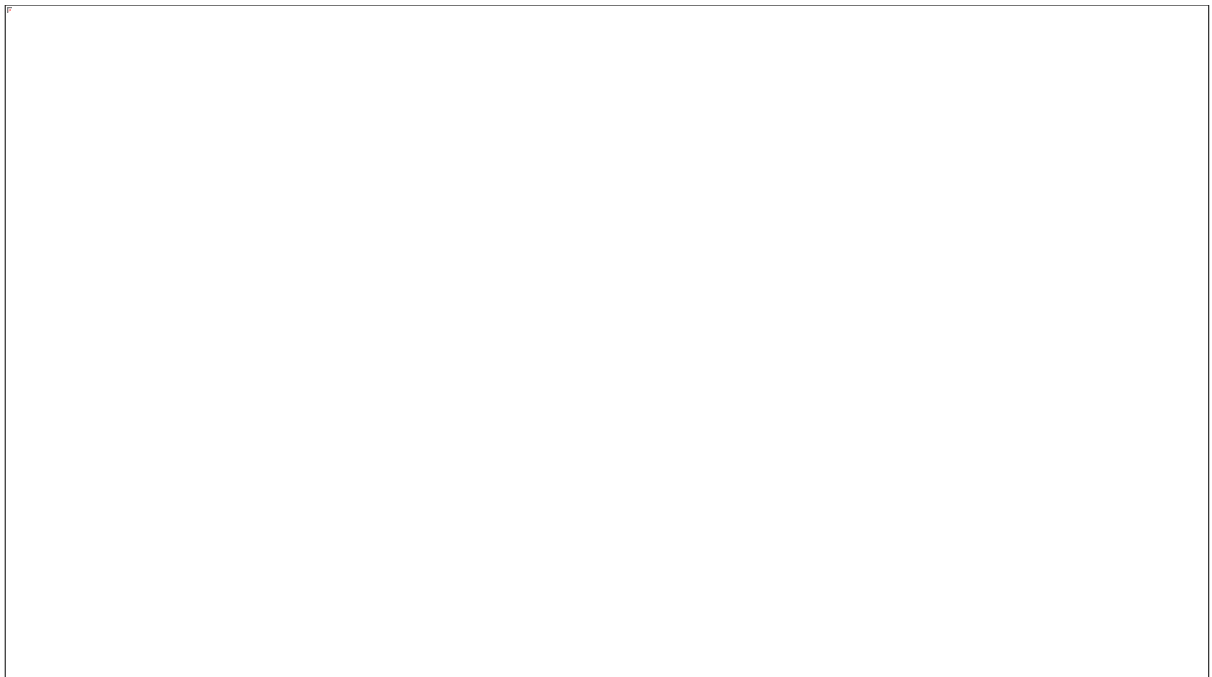


inserted the details in to the customer id



inserted a new record into the customers

<https://www.ons.gov.uk/businessindustryandtrade/itandinternetindustry/datasets/internetusers>



MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

SCHEMAS

- amazon
- amazon1
- amazon2
- dr
- hr1
- hr12
- hr7
- hr8
- hr9
- human_resources
- internet_users
 - Tables
 - Views
 - Stored Procedures
 - Functions
 - public_company_stods
 - rr

Administration Schemas

Information

Schema: internet_users

Query 1 Group Work* matthew gobbits code SQL File 4 SQL File 5*

```

1 CREATE DATABASE internet_users;
2 USE internet_users;
3
4

```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
48	15:32:35	CREATE TABLE 'internet_users`.`internet user1' ('Age group (years)' text, '2013' int, '2014' int, '2015' int, '2016' int, '2017' int, '2018' int, '2019' int, '2020' int)	OK	0.000 sec
49	15:32:35	PREPARE stmt FROM INSERT INTO 'internet_users`.`internet user1' ('Age group (years)', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020') VALUES ('16-24', 7075, 7074, 7155, 7129, 7036, 6992, 6877, 6844)	OK	0.000 sec
50	15:32:35	DEALLOCATE PREPARE stmt	OK	0.000 sec

Object Info Session

Type here to search

13°C Windy 15:32 30/04/2024

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Filter objects

SCHEMAS

- hr1
- hr12
- hr7
- hr8
- hr9
- human_resources
- internet_users
 - Tables
 - Views
 - Stored Procedures
 - Functions
 - public_company_stods
 - rr
 - sakila
 - subscriptions

Administration Schemas

Information

Table: internet user1

Columns:

- Age group (years) text
- 2013 int
- 2014 int
- 2015 int
- 2016 int
- 2017 int
- 2018 int
- 2019 int
- 2020 int

Query 1 Group Work* matthew gobbits code SQL File 4 SQL File 5* internet user1 x

```

1 SELECT * FROM internet_users.`internet user1`;

```

Result Grid

Age group (years)	2013	2014	2015	2016	2017	2018	2019	2020
Age group (years)	2013	2014	2015	2016	2017	2018	2019	2020
16-24	7075	7074	7155	7129	7036	6992	6877	6844
25-34	8457	8660	8582	8720	8815	8894	8895	8908
35-44	7952	7900	8053	8129	8118	8145	8243	8339
45-54	8005	8290	8498	8686	8803	8814	8810	8716
55-64	5821	6060	6361	6607	6888	7189	7495	7796
65-74	3562	3939	4390	4721	5031	5264	5339	5504
75+	1371	1534	1632	1925	2050	2262	2471	2933

Output

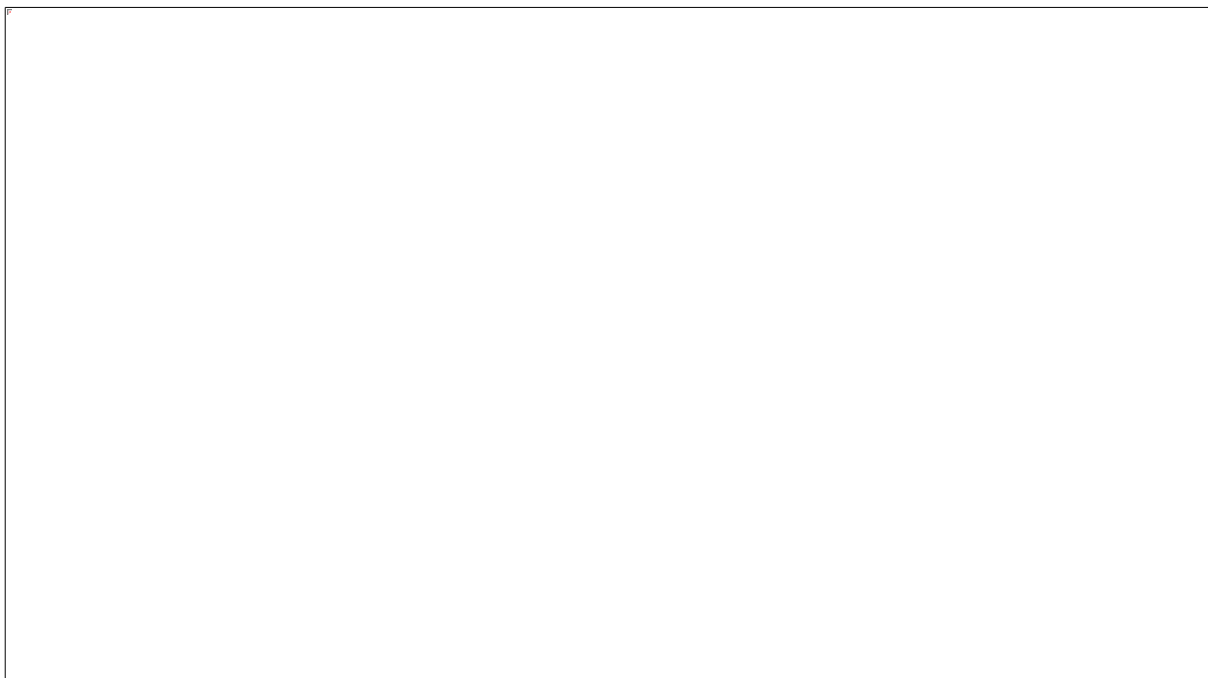
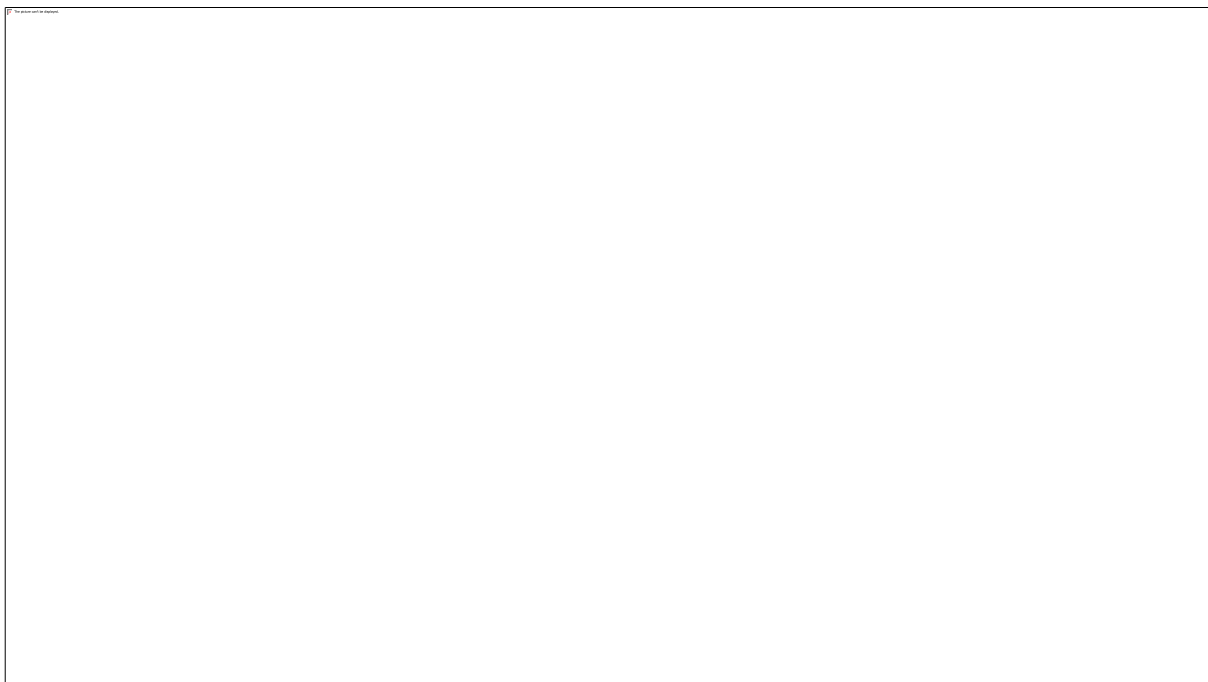
Action Output

#	Time	Action	Message	Duration / Fetch
51	15:32:35	SELECT * FROM internet_users.`internet user1`	OK	0.000 sec

Object Info Session

Type here to search

13°C Windy 15:35 30/04/2024



MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: 4.1 joins* x Group Work* SQL File 4 SQL File 5*

Limit to 1000 rows

```
22 SELECT c.*, o.*
23 FROM customers c
24 LEFT JOIN orders o
25 ON c.customer_id = o.customer_id
26 UNION
27 SELECT c.*, o.*
28 FROM customers c
29 RIGHT JOIN orders o
30 ON c.customer_id = o.customer_id;
```

Result Grid

	customer_id	customer_name	customer_email	order_id	customer_id	order_date	order_total
1	Alice	alice@example.com	102	1	01/02/2022	75	
1	Alice	alice@example.com	101	1	01/01/2022	50	
2	Bob	bob@example.com	103	2	15/02/2022	100	
3	Charlie	charlie@example.com	1023	1023	1023		
4	Dave	dave@example.com	104	4	01/03/2022	25	
5	Emily	emily@example.com	1023	1023	1023		

Result 4 x

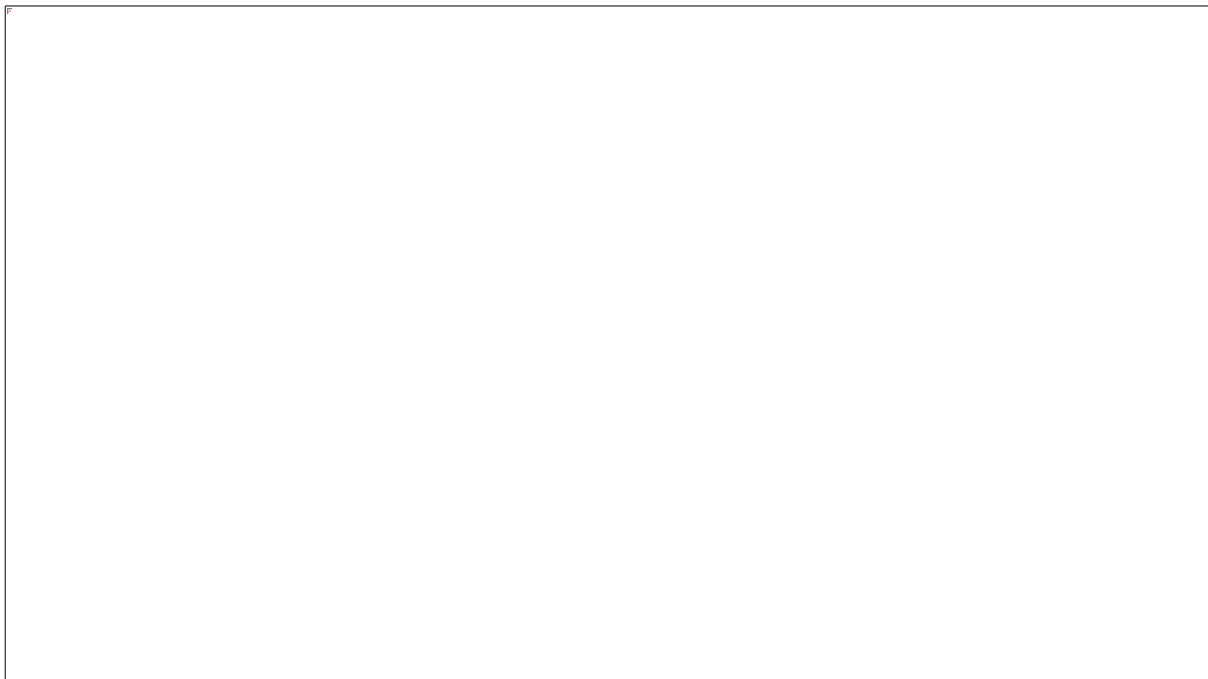
Output

#	Time	Action	Message	Duration / Fetch
19	14:07:37	SELECT c.*, o.* FROM customers c FULL OUTER JOIN orders o ON c.customer_id = o.cust...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds ...	0.000 sec
20	14:08:21	SELECT c.*, o.* FROM customers c LEFT JOIN orders o ON c.customer_id = o.customer_id ...	6 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Schema: hr9

11°C Cloudy 14:08 09/05/2024



PostgreSQL Portfolio, Using Pg Admin on Postgre SQL

= = select 1 column = =

SELECT *column name* FROM *table name*

= = select multiple columns = =

SELECT *column name 1, column name 2* FROM *table name*

= = select all columns = =

SELECT * FROM *table name*

= = get unique values on the column, no duplicates = =

SELECT DISTINCT *column name* FROM *table name*

= = count, report back rows in a column. After COUNT needs brackets = =

SELECT COUNT (*column name*) FROM *table name*

SELECT COUNT (*) FROM *table name*

SELECT (COUNT *column name*) FROM *table name*

= = count of distinct values in column = =

SELECT COUNT (DISTINCT *column name*) FROM *table name*

= = total number of rows = =

SELECT COUNT (*) FROM *table name*

== WHERE with conditional operators ==

```
SELECT column 1, column2
FROM table name
WHERE condition;
```

=

>

<

>=

<=

< > or !=

AND

NOT

== example of conditional operators==

```
SELECT column 1, column2
FROM table name
WHERE condition1 = thing searching for
```

== another example of conditional operators==

```
SELECT column 1, column2
FROM table name
WHERE condition1 = thing searching for AND condition1 = thing searching;
```

== selecting a specific value from a column ==

```
SELECT *FROM table name
WHERE column name = value
```

== selecting a specific value from a column ==

```
SELECT *FROM table name
WHERE column name1 > value
AND column name2 >= value
```

== count of a selection = =

```
SELECT COUNT (column1 or *) FROM table name
WHERE column2 > value
AND column 3>= value
```

== using OR count of a selection = =

```
SELECT *FROM table name
WHERE column1 = value1 OR column1 = value2
```

== using NOT ==

```
SELECT * FROM table name  
WHERE column != value
```

==ORDER BY==

```
SELECT column1, column2  
FROM table  
ORDER BY column1 ASC or DESC
```

== order by column1 first then column2, usually ascending order ==

```
SELECT column1, column2, column3  
FROM table  
ORDER BY column1, column2
```

== order by column1 first then column2, descending and then ascending order ==

```
SELECT column1, column2, column3  
FROM table  
ORDER BY column1 DESC, column2 ASC;
```

== LIMIT, limit rows returned==

```
SELECT * FROM table  
ORDER BY column1 ASC  
LIMIT number of rows (this gives a limited number of rows)
```

```
SELECT * FROM table  
ORDER BY column1 ASC (or DESC)  
LIMIT number of rows
```

BETWEEN

== operator used to match a value against a range of values BETWEEN low and high==

Value >= low AND value <= high

== values BETWEEN low AND high==

== DATES YYYY-MM-DD ==

DATES BETWEEN date1 AND date2

== Betweens==

```
SELECT * FROM table  
WHERE column1 BETWEEN figure1 AND figure2;
```

SELECT * FROM table
WHERE column1 NOT BETWEEN figure1 AND figure2;

SELECT * FROM table
WHERE column1 BETWEEN 'date1' AND 'date';

==IN==
== check to see if value is IN a list of multiple options==

SELECT column FROM table
WHERE column IN ('value1', 'value2');

SELECT column FROM table
WHERE column NOT IN ('value1', 'value2');

== gives the count of rows including values or NOT==
SELECT COUNT (*) FROM table
WHERE column IN ('value1', 'value2');

SELECT COUNT (*) FROM table
WHERE column NOT IN ('value1', 'value2');

== LIKE and ILIKE==
== wild card % '_ %' '%_ %' '%_ '
== like is case sensitive

For example a word beginning with A
WHERE column LIKE 'A%'

For example a word ending with A
WHERE column LIKE '%a'

== underscore allows to replace a single character==
== like is case sensitive==

WHERE *column* LIKE 'Billio_'
WHERE column LIKE 'word_'

SELECT * FROM table
WHERE column LIKE '_ %';

SELECT COUNT* FROM table
WHERE column LIKE '_ %';

SELECT COUNT* FROM table
WHERE column1 LIKE '_ %' AND column2 LIKE '_ %';

== AGGREGATION FUNCTIONS==

== Use round () for rounding to decimal places ==

AVG ()

COUNT()

MAX()

MIN()

SUM()

Only in SELECT or HAVING

EXAMPLES

SELECT* FROM table;

SELECT MIN (column name) FROM table;

SELECT MAX (column name) FROM table;

SELECT MAX (column name), MIN (column name) FROM table;

==count==

SELECT COUNT (column name) FROM table;

==AVG==

SELECT AVG (column name) FROM table;

== AVG, how to round up ==

SELECT AVG ((column name), decimal places) FROM table;

==SUM==

SELECT SUM (column name) FROM table;

== GROUP BY==

== allows us to aggregate (AGG) columns per some category==

== categorical needed to group by, categorical are not continuous ==

SELECT category column, AGG (data column) *AGG/SUM/AVG/COUNT etc*

FROM table

GROUP BY category column;

== Group by clause must appear right after a FROM or WHERE statement==

```
SELECT category column, AGG (data column)
FROM table
WHERE category column != 'data in columns'
GROUP BY category column
```

```
SELECT category column, AGG (data column)
FROM table
WHERE category column != 'data in columns'
GROUP BY category column
```

== filter out divisions before WHERE or HAVING ==

== GROUP BY==

```
SELECT column FROM table
GROUP BY column
ORDER BY column
```

```
SELECT column1, SUM(column2) FROM table
GROUP BY column1
ORDER BY SUM(column2) DESC      or ASC
```

```
SELECT column1, COUNT(column2) FROM table
GROUP BY column1
ORDER BY COUNT(column2) DESC    or ASC
```

```
SELECT column1, column2, SUM(column3) FROM table
GROUP BY column1, column 2
ORDER BY column1
```

```
SELECT DATE (column1) FROM table
GROUP BY DATE (column1)
```

```
SELECT DATE (column1), SUM(column2) FROM table
GROUP BY DATE (column1)
ORDER BY SUM (column2)  DESC
```

==example==

```
SELECT DATE (sales date), SUM(amount of sales) FROM table
GROUP BY DATE (sales date)
ORDER BY SUM (amount of sales)  DESC or ASC
```

==example==

```
SELECT staff_id COUNT (*)  
FROM employment table  
GROUP BY staff_id
```

==example==

```
SELECT rating_column, AVG cost_column  
FROM film table  
GROUP BY rating column
```

==example==

```
SELECT customer_id, SUM (amount)  
FROM payment column  
GROUP BY customer_id column  
ORDER BY SUM(amount column) DESC  
LIMIT 5
```

HAVING

==HAVING==

== filter after an aggregation taken place==

==having allows to aggregate result as a filter along with GROUP BY==

```
SELECT column1, SUM (column2)  
FROM table  
GROUP BY column1  
HAVING SUM (column2) >, >=, <=, <, != VALUE
```

AS alias

==AS==

```
SELECT column AS alias or new name  
FROM table
```

== The AS operator gets used /executed at the very end of a query, meaning we cannot use the alias inside a WHERE operator.

```
SELECT column1, SUM column2  
FROM table  
GROUP BY column1
```

```
SELECT column1, SUM column2 AS new name/alias  
FROM table  
GROUP BY column1
```

```
SELECT column1, column2 AS new name
FROM table
WHERE column2 > number
```

INNER JOINS

== Joins combine multiple tables together==

== Joins help decide how to deal with information only presented in one of the joined tables

== An INNER JOIN will match results with the set of records that match in both tables==

```
SELECT *FROM TABLE A
INNER JOIN TABLE B
ON TABLEA.col_match = TABLEB.col_match
```

The same as above

```
SELECT *FROM TABLE B
INNER JOIN TABLE A
ON TABLEA.col_match = TABLEB.col_match
```

Use this type of formula for inner joins

```
INNER JOIN
SELECT*FROM table1
INNER JOIN table2
ON table1.mutual column = table2.mutual column
```

Selects matches from both tables

Symmetrical

REMOVE DUPLICATES from inner join

```
SELECT other column from table1, table2.mutual column. Other column from table 2
FROM table 1
INNER JOIN table 2
ON table1.mutual column = table2.mutual column
```

INNER JOIN

```
SELECT column1, table. Column 2 (the mutual column), column 3
FROM table 1
INNER JOIN table2
ON table1. Column 2 (mutual column) = table2. Column2 (mutual column)
```

ONLY SHOWS RECORDS SHOWN BY BOTH

FULL OUTER JOINS

== deal with values only present in one of the tables being joined ==

==full outer==

==left outer==

==right outer==

```
SELECT * FROM table1
FULL OUTER JOIN table2
ON table1.column same as other table = table2.column same as other table
```

```
SELECT * FROM table1
FULL OUTER JOIN table2
ON table1.mutual column = table2. mutual column
```

FULL OUTER JOIN with WHERE

== get rows unique to either table (rows not found in both tables) ==

This shows results that are unique to table 1 and table 2 opposite to inner joins

```
SELECT * FROM table1
FULL OUTER JOIN table2
ON table1.mutual column = table2.mutual column
WHERE table1.column unique IS NULL
OR table2.column unique IS NULL
```

Column unique usually is ID

See how many distinct/unique rows there are

```
SELECT COUNT (DISTINCT column)
FROM table
```

LEFT OUTER JOIN

Set of records in the left table if there is no match with the right table the results are null

```
SELECT * FROM table1
LEFT OUTER JOIN table2
ON table1.mutual matching column = table2.mutual matching column
```

THE ORDER OF THE TABLE MATTERS

Rows found in table 1 but not in table 2

```
SELECT *FROM table1
LEFT OUTER JOIN table2
ON table1.mutual column = table2.mutual column
WHERE table2.unique IS MULL
```


Unique column = column in question, usually ID

RIGHT OUTER JOIN

Same as left join but opposite

```
SELECT * FROM Table1
RIGHT OUTER JOIN table2
ON table1.matching column = table2.matching column
```

```
SELECT * FROM Table1
RIGHT OUTER JOIN table2
ON table1.matching column = table2.matching column
WHERE table1.unique column IS NULL
```

UNIONS

==unions combine the results of 2 or more SELECT statements==
==concatenate 2 results together==

```
SELECT columns FROM table1
UNION
SELECT columns FROM table2
```

```
SELECT columns FROM table1
UNION
SELECT columns FROM table2
ORDER BY column;
```

Timestamps and Extract

Displaying current information
Report back time and date information
Time- contains only time
Date – contains only date
Timestamp – contains date and time
Time stamp tz – contains date, time and timezone

```
TIMEZONE
NOW
TIME OF DAY
CURRENT_TIME
CURRENT_DATE
```

SHOW ALL (shows parameters)
SELECT NOW () = shows time information
SELECT TIMEOFDAY () = shows time of day in string form
SELECT CURRENT TIME = current time zone
SELECT CURRENT_DATE=

TIMESTAMPS AND EXTRAC PART 2

EXTRACT ()
AGE ()
TO_CHAR ()

EXTRACT ()
- YEAR
- MONTH
- DAY
- WEEK
- QUARTER

EXTRACT (YEAR FROM column with date)

AGE (date column)
13 years 1 month 5 days 01:34:13.003423 hours: min: seconds

TO_CHAR(date column, 'mm-dd-yyyy')

SELECT EXTRACT (YEAR FROM column) AS code/alias
FROM table

SELECT AGE (column) FROM table

SELECT TO_CHAR (column, 'month yyyy') choose how you want the date on table

Timestamps and Extract Challenge Tasks

SELECT
DISTINCT (TO_CHAR(column, 'MONTH')) *YEAR, QUARTER etc instead of MONTH*
FROM table

MATHEMATICAL FUNCTIONS

SELECT * FROM table

SELECT column1/column2 FROM table

SELECT ROUND(column1/column2,2) FROM table

SELECT ROUND (column1/column2,2) *100 AS alias
FROM TABLE

STRING FUNCTIONS & OPERATIONS

Google this to see the list of functions 'string function and operators'

SELECT LENGTH (column) FROM table

This shows length of words in that column

SELECT column1 || ' space' || column2 FROM table

This does concatenates

SELECT column1 || ' space' || column2 AS alias FROM table

This does concatenates with alias

SELECT UPPER column1 || ' space' || column2 AS alias FROM table

This does concatenates with alias in capital letters

Example of creating an email

SELECT

LOWER (LEFT (FIRST_NAME,1)) || lower (LAST_NAME) || '@gmail.com'

AS EMAIL

FROM table

SUBQUERY

Complex queries creating a query on results of another query

SELECT column FROM table

SELECT AVG column FROM table

SELECT column1, column2

FROM table

WHERE column > (SELECT AVG(column2)

FROM table

EXIST

```
SELECT column  
FROM table  
WHERE EXISTS  
(SELECT column FROM table WHERE condition)
```

IN

```
SELECT column1, column2  
FROM table1  
WHERE column1 IN  
(SELECT column1 FROM table2)
```

IN Example

```
SELECT column1, column2  
FROM table1  
WHERE column1 IN  
(SELECT _.column1 FROM table2  
INNER JOIN _ ON_.column3  
= table2.column3  
WHERE column4 BETWEEN _ AND_ ORDER BY column2)
```

EXISTS

```
SELECTS column1, column2  
FROM table1 AS alias1  
WHERE EXISTS  
(SELECT * FROM table2 AS alias2  
WHERE alias2.column3 = alias1.column3  
AND column 4 > value)
```

You can get a NOT by using NOT EXISTS

SELF JOINS

- query in which table is joined to itself
- comparing values in a column of rows within the same table

```
SELECT tableA.col, tableB.col  
FROM table AS table A  
JOIN table AS table B ON  
tableA.same_col = tableB.other_col
```

example

```
SELECT emp.name, report.name AS rep  
FROM employees AS emp  
JOIN employees AS report ON  
Emp.emp_id = report.report_id
```

Example

```
SELECT f1.title, f2.title, f1 length  
FROM film AS f1  
INNER JOIN film AS f2 ON  
F1.film_id != f2.film_id  
AND f1.length = f2.length
```

CREATING DATABASES AND TABLES

DATA TYPES

Boolean – true or false

Character – char, varchar , text

Numeric – integer, floating-point number

Temporal – date, time, timestamp, interval

PRIMARY and FOREIGN KEYS

Primary key is a column or a column or a group of column used to identify a row uniquely in a table

Not null, means must be entry

Allows one to know what columns to join table together

A foreign key is a field or a group of fields in a table that uniquely identifies a row in another table.

A foreign key is defined in a table that references to the primary key of the other table

Table that contains foreign key is the referencing table

A table can have multiple foreign keys depending on its relationship with other tables

PK in a column means primary key

Foreign keys are constraints

CONSTRAINTS

Rules enforced on data columns on table

NOT NULL is a constraint

UNIQUE is a constraint, all values in a column are different

PRIMARY KEY

FOREIGN KEY

CHECK - ensure all values satisfy

EXCLUSION

REFERENCES

CREATE TABLE

CREATE DATABASE (use side bar to do so)

```
CREATE TABLE table_name(  
  Column_name_1 DATA TYPE, CONSTRAINT,  
  Column_name_2 DATA TYPE, CONSTRAINT,  
)
```

INSERT INTO TABLE

SELECT* FROM table_name

INSERT INTO table_name (column1, column2, column3, column4)

VALUES

(, , ,)

UPDATE

-allows for changes in columns in table

UPDATE table

SET column1 = value1

Column2 = value 2

WHERE condition;

Examples

-allows for changes in columns in table

UPDATE account

SET last_login = CURRENT_TIME STAMP

WHERE last_login is NULL

Examples

UPDATE account

SET last_login = created on

Updated joins

UPDATE Table A

SET original_col = tableB. New_col

FROM Table B

WHERE tableA.id = TableB.id

Return affected rows

UPDATE account

SET last_login = created_on

RETURNING account_id, last_login

DELETE

DELETE FROM table

WHERE row_label = value

DELETE FROM tableA

USING tableB

WHERE tableA.id = tableB.id

DELETE FROM table

ALTER TABLE

ALTER TABLE table_name

ADD COLUMN new_column TYPE

ALTER TABLE table_name

DROP COLUMN column_name

ALTER TABLE table_name

ALTER COLUMN column_name

DROP DEFAULT

DROP
SET NOT NULL
SET DEFAULT VALUE
DROP NOT NULL
ADD CONSTRAINT constratin_name

DROP TABLE

ALTER TABLE table_name
DROP COLUMN column_name CASCADE

ALTER TABLE table_name
DROP COLUMN column_1
DROP COLUMN column_2

CHECK CONSTRAINT

Example

```
CREATE TABLE example(  
  Ex_id SERIAL PRIMARY KEY  
  Age SMALL INT CHECK (age>21)  
  Parent_age SMALL INT CHECK (  
    parent_age> age)  
);
```

Example

```
CREATE TABLE employees(  
  Emp_id SERIAL PRIMARY KEY,  
  First_name VARCHAR (50) NOT NULL,  
  Last_name VARCHAR (50) NOT NULL,  
  Birthdate DATE CHECK (birthday> '1900,01,01'),  
  Hire_date DATE CHECK (hiredate>birthdate),  
  Salary INTEGER CHECK (salary>0),
```

Then you add the values to the SQL , see if they are in the constraint.

Conditional Expression and Procedures

Case statement

- General CASE
- CASE expression

CASE

```
WHEN condition1 THEN result1  
WHEN condition2 THEN result 2  
ELSE some_other_result  
END
```

```
SELECT*FROM test;  
SELECT a,  
CASE WHEN a = 1 THEN 'one'  
      WHEN a= 2 THEN 'two'  
ELSE 'other'  
END  
FROM test;
```

CASE expression

First evaluates an expression then compares the result with each value in the WHEN clauses sequentially

CASE expression

```
WHEN value1 THEN result 1  
WHEN value2 THEN result 2  
ELSE come_other_result  
END
```

Example

```
SELECT a  
CASE a WHEN 1 THEN 'one'  
      WHEN 2 THEN 'two'  
ELSE 'other'  
END  
FROM test;
```

CASE Example

```
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
```

CASE Expression

```
SELECT customer_id.  
CASE customer_id.  
WHEN 2 THEN 'winner'  
WHEN 5 THEN 'second place'  
ELSE 'Normal'  
END AS raffle_results  
FROM customer
```

Case expression example

```
SELECT * FROM film  
SELECT rental_rate FROM film  
SELECT rental_rate  
CASE rental_rate  
WHEN 0.99 THEN 1  
ELSE 0  
END  
FROM film
```

COALESCE

- Coalesce function returns first arguments that is not null. If all arguments are null the COALESCE function will return null

COALESCE(ARG-1, ARG-2.....ARF-N)

```
SELECT COALESCE (1,2)  
SELECT COALESCE (NULL,2,3)  
2
```

Useful when querying a table that contains null values and substituting it with another value

Example: what is the final price

```
SELECT item, (price_discount) AS final  
FROM table  
Doesn't work for item B
```

So

```
SELECT item (price-COALESCE (discount, 0))  
AS final FROM table
```

No need to make changes to table

CAST

Convert one data type into another

Not every data type can be cost into another data type

Syntax

```
SELECT CAST ('5' AS INTEGER)
```

PostgreSQL CAST operator

```
SELECT CAST '5' :: INTEGER
```

Use this in a SELECT query with a column name instead of a single instance

```
SELECT CAST (date AS TIMESTAMP) FROM table
```

```
SELECT CAST ('5' AS INTEGER) AS new int
```

```
SELECT '5':: INTEGER
```

```
SELECT CAST (inventory_id AS VARCHAR) FROM table
```

```
SELECT CHAR_LENGTH (CAST(inventory-id AS VARCHAR) FROM table
```

NULL IF

NULL IF (arg1, ar2)

Returns NULL if both inputs equal

Postgre > create > new database

Gives Ratio Example

```
SELECT (  
SUM (CASE WHEN department = 'A' THEN 1  
ELSE 0 END)/  
SUM (CASE WHEN department = 'B' THEN 1  
ELSE 0 END)  
AS department ratio
```

```
DELETE FROM depts  
WHERE department = 'B'  
SELECT*FROM depts
```

```
NULL IF  
SELECT (  
SUM (CASE WHEN department='A' THEN 1 ELSE 0 END)/  
NULLIF (SUM(CASE WHEN department = 'B' THEN 1  
ELSE 0 END),0)  
)
```

```
AS department ratio  
FROM depts
```

Check if something is equal to zero

VIEWS

View to quickly see the query

```
CREATE VIEW customer_info AS
```

```
SELECT first_name, last_name, address FROM customer  
INNER JOIN address  
ON customer.address_id = address.address_id
```

```
SELECT* FROM customer_info  
Gives same output as the join
```

```
CREATE or REPLACE VIEW customer_info AS  
NEW JOIN (but new version)
```

```
DROP VIEW IF EXISTS customer_info
```

```
ALTER VIEW customer_info RENAME TO customer_info
```

IMPORT CSV files

Tables> simple> import/export

Select import or export

Select columns

Select file name

Select file location

Format csv

