

EXAM

Task1

Customer table

Tables (3 items)

Add Table Customer Manager Branch

Views (0 items)

Add View

Routines (0 items)

Customer - Table

Table Name: Schema: **mydb**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
CustomerID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
BranchCustID	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Foreign key for customer table

Customer - Table

Table Name: Schema: **mydb**

Foreign Key Name	Referenced Table	Column	Referenced Column
CustomerBranch	`mydb`.`Branch`	<input type="checkbox"/> CustomerID <input type="checkbox"/> Name <input checked="" type="checkbox"/> BranchCustID	BranchID

Manager table

Add View

Routines (0 items)

Manager - Table

Table Name: Schema: **mydb**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ManagerID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
BranchManID	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type:

Foreign key for manager table

Routines (0 items)

Manager - Table

Table Name: Schema: **mydb**

Foreign Key Name	Referenced Table	Column	Referenced Column
ManagerBranch	`mydb`.`Branch`	<input type="checkbox"/> ManagerID <input type="checkbox"/> Name <input checked="" type="checkbox"/> BranchManID	BranchID

Foreign Key Options

On Update: **NO ACTION**

On Delete: **NO ACTION**

☐ Skip in SQL generation

Foreign Key Comment:

Branch table

Add View

Routines (0 items)

Brancht - Table

Table Name: Schema: **mydb**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
BranchID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
BranchName	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Location	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:

Charset/Collation:

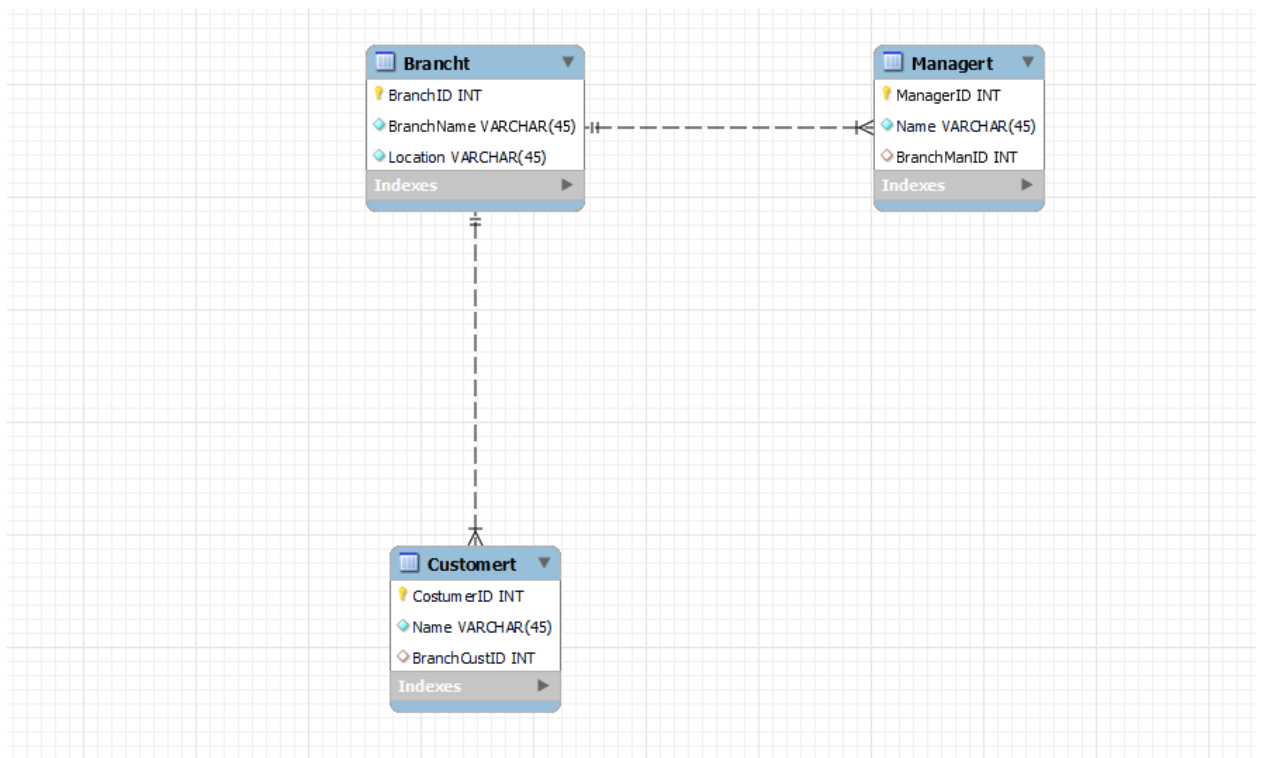
Comments:

Data Type:

Default:

Storage: ☐ Virtual ☐ Stored

Build a diagram



Task 2

Create table by sql and assign PK

```

1 • create table Customer(
2     CustomerID int not null,
3     CustomerName varchar(45) not null,
4     CustomerLastName varchar(45) not null,
5     CustomerEmail char(45) not null,
6     CONSTRAINT CustomerID_PK PRIMARY KEY(CustomerID)
7 );
8

```

Create a manager table and set pk

```

• create table Manager(
    ManagerID int not null,
    ManagerName varchar(45) not null,
    ManagerLastName varchar(45) not null ,
    CONSTRAINT ManagerID_PK PRIMARY KEY(ManagerID)
)

```

Insert values to customer

```

Limit to 50 rows
1 • insert into customer values(1,"Arina", "Sheredekina", "arina.s@something.com");
2 • insert into customer values(2,"K", "Sheredekina", "k.s@something.com");
3 • insert into customer values(3,"S", "Sheredekina", "s.s@something.com");
4 • insert into customer values(4,"D", "Sheredekina", "d.s@something.com");

```

1 • `SELECT * FROM finalprep.customer;`

<

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

	CustomerID	CustomerName	CustomerLastName	CustomerEmail
▶	1	Arina	Sheredekina	arina.s@something.com
	2	K	Sheredekina	k.s@something.com
	3	S	Sheredekina	s.s@something.com
	4	D	Sheredekina	d.s@something.com
*	NULL	NULL	NULL	NULL

Insert for manager

Limit to 50 rows

1 • `insert into manager values(1,"A","S");`
 2 • `insert into manager values(2,"B","S");`
 3 • `insert into manager values(3,"C","S");`
 4 • `insert into manager values(4,"D","S");`

finalprep - Schema SQL File 2* SQL File 3* customer SQL File 4* **manager** x

Limit to 50 rows

```
1 • SELECT * FROM finalprep.manager;
```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Con

	ManagerID	ManagerName	ManagerLastName
▶	1	A	S
	2	B	S
r(45)	3	C	S
r(45)	4	D	S
*	NULL	NULL	NULL

Import csv file

SCHEMAS

Filter objects

- finalprep
 - Tables
 - customer
 - manager
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - vgsalescopy
 - Views

Administration Schemas

Information

Table: vgsalescopy

Columns:

n»iRank	int
Name	text
Platform	int
Year	int
Genre	text
Publisher	text
NA_Sales	double
EU_Sales	double
JP_Sales	int
Other_Sales	double
Global_Sales	double

Output

Action

Change column



Navigator

SCHEMAS

Filter objects

finalprep

- Tables
 - customer
 - manager
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - vgsalescopy
- Views

Administration Schemas

Information

Table: vgsalescopy

Columns:

Column	Data Type	PK
Rank1	char(10)	PK
Name	text	
Platform	int	
Year	int	
Genre	text	

SQL File 5* **SQL File 6***

Limit to 50 rows

```

1 • ALTER table vgsalescopy
2   add constraint Rank1_PK PRIMARY KEY(Rank1);
  
```

Create two indexes

Navigator

SCHEMAS

Filter objects

finalprep

- Tables
 - customer
 - manager
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - vgsalescopy
- Views

Administration Schemas

Information

Table: vgsalescopy

Columns:

Column	Data Type	PK
Rank1	char(10)	PK
Name	text	
Platform	int	
Year	int	

SQL File 5* **SQL File 6*** **SQL File 7***

Limit to 50 rows

```

1 • create index index1 on vgsalescopy(Year);
2 • create index index2 on vgsalescopy(Platform);
  
```

Indexes in Table					Index Details	
Visible	Key	Type	Uni...	Columns	Key Name:	Packed:
<input checked="" type="checkbox"/>	PRIMARY	BTREE	YES	Rank1	Index Type:	Unique:
<input checked="" type="checkbox"/>	index1	BTREE	NO	Year	Allows NULL:	
<input checked="" type="checkbox"/>	index2	BTREE	NO	Platform	Cardinality:	
					Comment:	
					User Comment:	

Columns in table

Task 3

Create role

1 create role adminadmin;

<

Output

Action Output

	#	Time	Action
✓	22	11:57:13	ALTER table vgsalescopy add constraint Rank1_PK PRIM.
✓	23	11:58:40	create index index1 on vgsalescopy(Year)
✓	24	11:58:40	create index index2 on vgsalescopy(Platform)
✗	25	12:01:50	create role adminadmin
✓	26	12:02:02	drop role adminadmin
✓	27	12:02:14	create role adminadmin

Grant permissions to role on file

- 1 • `use finalprep;`
- 2 • `grant select, update, delete on vgsalescopy to adminadmin;`

Output

#	Time	Action
26	12:02:02	drop role adminadmin
27	12:02:14	create role adminadmin
28	12:04:44	use finalprep
29	12:04:44	grant select, update, delete on vgsalescopy to adminadmin
30	12:05:42	use finalprep
31	12:05:42	grant select, update, delete on vgsalescopy to adminadmin

Create user with the role

```
SQL File 8  SQL File 9  SQL File 10  x
1  create user "Arina" identified by "123" default role adminadmin;
```

Select command

Task 4

Create 2 sp

```
1 DELIMITER //
```

```
2 • create PROCEDURE sp1()
```

```
3   Begin
```

```
4     select Rank1, Name, Year from vgsalescopy;
```

```
5   End//
```

```
6 DELIMITER //
```

```
1 DELIMITER //
```

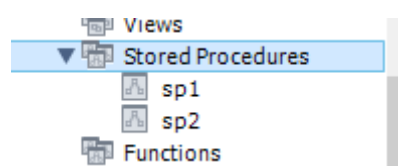
```
2 • create PROCEDURE sp2()
```

```
3   begin
```









```
4     select Global_Sales from vgsalescopy where Global_Sales>0.5;
```

```
5   end//
```


```
6 DELIMITER //
```



SQL File 11*SQL File 12*sp1sp2



Limit to 50 rows





1 •


call finalprep.sp1();

2

Result Grid

 Filter Rows:

Export: 

Wrap Cell Content: 

	Rank1	Name	Year
▶	1108	Ms. Pac-Man	1981
	1155	River Raid	1981
	1308	Donkey Kong	1981
	1431	Centipede	1981
	1558	Atlantis	1981
	1768	Kaboom!	1980
	1850	Manamania	1981

The screenshot shows a SQL IDE with a file explorer on the left and a main editor area. The editor has tabs for 'SQL File 11*', 'SQL File 12*', 'sp1', and 'sp2'. The 'sp2' tab is active, showing a SQL script with two lines: `1 call finalprep.sp2();` and `2`. Below the editor, there is a 'Result Grid' section. It has a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The grid displays a single column named 'Global_Sales' with six rows of values: 1.65, 1.6, 1.46, 1.36, 1.27, and 1.15. Below the grid is a 'Result 1' tab and an 'Output' section with an 'Action Output' dropdown.

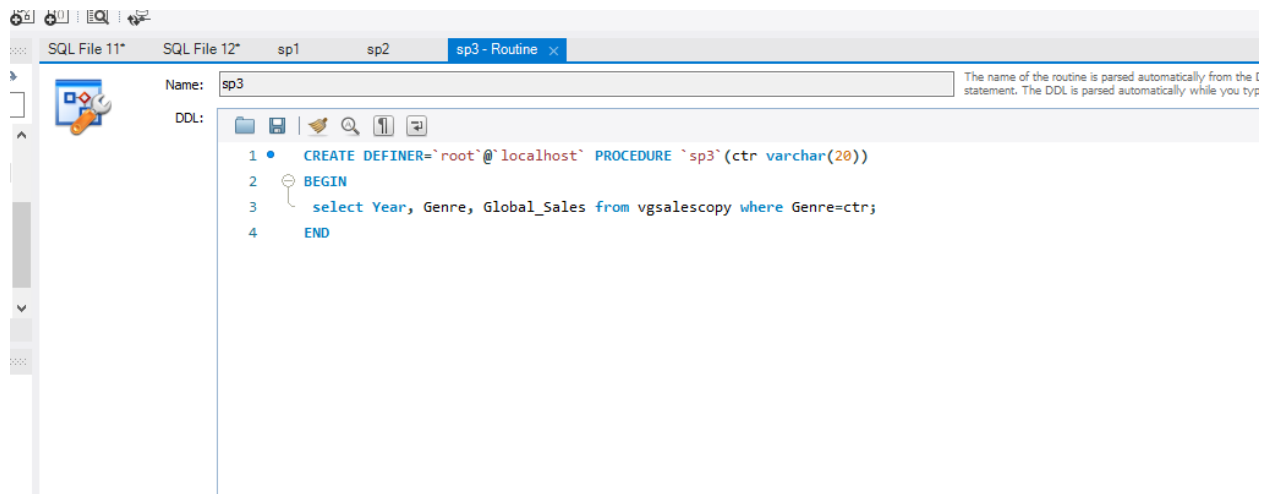
Global_Sales
1.65
1.6
1.46
1.36
1.27
1.15

With parameter

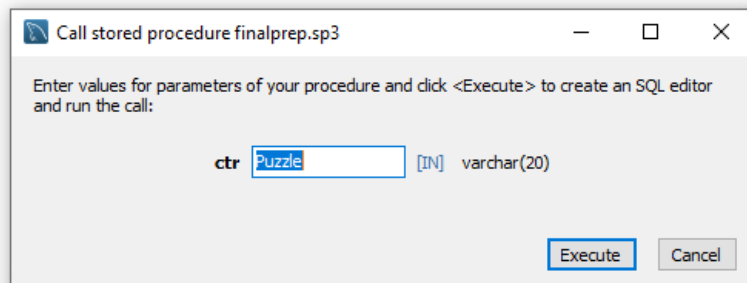
The screenshot shows a SQL IDE with a file explorer on the left and a main editor area. The editor has tabs for 'SQL File 11*', 'SQL File 12*', 'sp1', 'sp2', and 'new_procedure - Routine'. The 'new_procedure - Routine' tab is active, showing a SQL script for creating a procedure. The script is as follows:

```
1 CREATE PROCEDURE sp3 (ctr varchar(20))
2 BEGIN
3 select Year, Genre, Global_Sales from vgsalescopy where Genre=ctr;
4 END
5
```

Below the script, there is a 'Name' field containing 'new_procedure' and a 'DDL' section. The 'DDL' section has a 'Name' field containing 'new_procedure' and a 'DDL' section. The 'DDL' section has a 'Name' field containing 'new_procedure' and a 'DDL' section.



```
- select Year, Genre, Global_Sales from vgsalescopy where Genre=ctr;
END
```



SQL File 11* SQL File 12* sp1 sp2 sp3 - Routine **sp3** ×

Limit to 50 rows

```
1 • call finalprep.sp3('Puzzle');
2
```

<

Result Grid Filter Rows: Export: Wrap Cell Content: [IA](#)

	Year	Genre	Global_Sales
▶	1981	Puzzle	1.65
	1981	Puzzle	0.59

Limit to 50 rows

```
1 create View View1 as
2 select Year, Name, Platform from vgsalescopy where Global_Sales>0.5;
```


The screenshot shows a SQL IDE interface with a toolbar at the top containing icons for file operations, execution, and search. The main editor displays a single SQL query: `SELECT * FROM finalprep.view1;`. Below the editor, the 'Result Grid' tab is active, showing a table with three columns: 'Year', 'Name', and 'Platform'. The table contains five rows of data. Below the result grid, there are tabs for 'view1 1' and 'Output', and a section for 'Action Output'.

Year	Name	Platform
1981	Ms. Pac-Man	2600
1981	River Raid	2600
1981	Donkey Kong	2600
1981	Centipede	2600
1981	Atlantis	2600

Trigger

The screenshot shows a SQL IDE interface with a toolbar at the top. The main editor displays a SQL query to create a trigger. The query is as follows:

```
1 DELIMITER //  
2 • create Trigger before_insert_customer  
3 before insert on customer for each row  
4 begin  
5 if new.CustomerName="Arina" then set new.CustomerName="Adrian";  
6 end if;  
7 end//  
8 DELIMITER //
```

SQL File 11* SQL File 12* SQL File 13* view1 SQL File 14* SQL File 15* x

Limit to 50 rows

1 • show triggers;

metLastI
merEmail

sys

insert

>

mer

Result Grid

Trigger	Event	Table	Statement	Timing	Created	sql_mode
before_insert_customer	INSERT	customer	begin if new.CustomerName="Arina" then set ...	BEFORE	2024-12-09 12:36:35.54	ONLY_FULL_GROUP

Result 1 x

Output

Action Output

#	Time	Action	Message
39	12:24:02	call finalprep.sp3('Puzzle')	2 row(s) returned
40	12:24:21	call finalprep.sp3('Puzzle')	2 row(s) returned
41	12:26:15	create View View1 as select Year, Name, Platform from vgsalescopy where Global_Sales>0.5	0 row(s) affected
42	12:27:09	SELECT * FROM finalprep.view1 LIMIT 0, 50	26 row(s) returned
43	12:36:35	create Trigger before_insert_customer before insert on customer for each row begin if new.CustomerName="Arina" t...	0 row(s) affected
44	12:36:59	show triggers	1 row(s) returned

2	K	Sheredekina	k.s@something.com
3	S	Sheredekina	s.s@something.com
4	D	Sheredekina	d.s@something.com
5	Arina	Pina	h.s@something.com
NULL	NULL	NULL	NULL

CustomerID	CustomerName	CustomerLastName	CustomerEmail
1	Arina	Sheredekina	arina.s@something.com
2	K	Sheredekina	k.s@something.com
3	S	Sheredekina	s.s@something.com
4	D	Sheredekina	d.s@something.com
5	Adrian	Pina	d.s@something.com
NULL	NULL	NULL	NULL

customer 1 x