

LATIHAN PERTEMUAN 11
PRAKTIKUM PEMOGRAMAN BERBASIS WEB
Untuk Memenuhi Praktikum Pemograman Berbasis Web



Oleh:

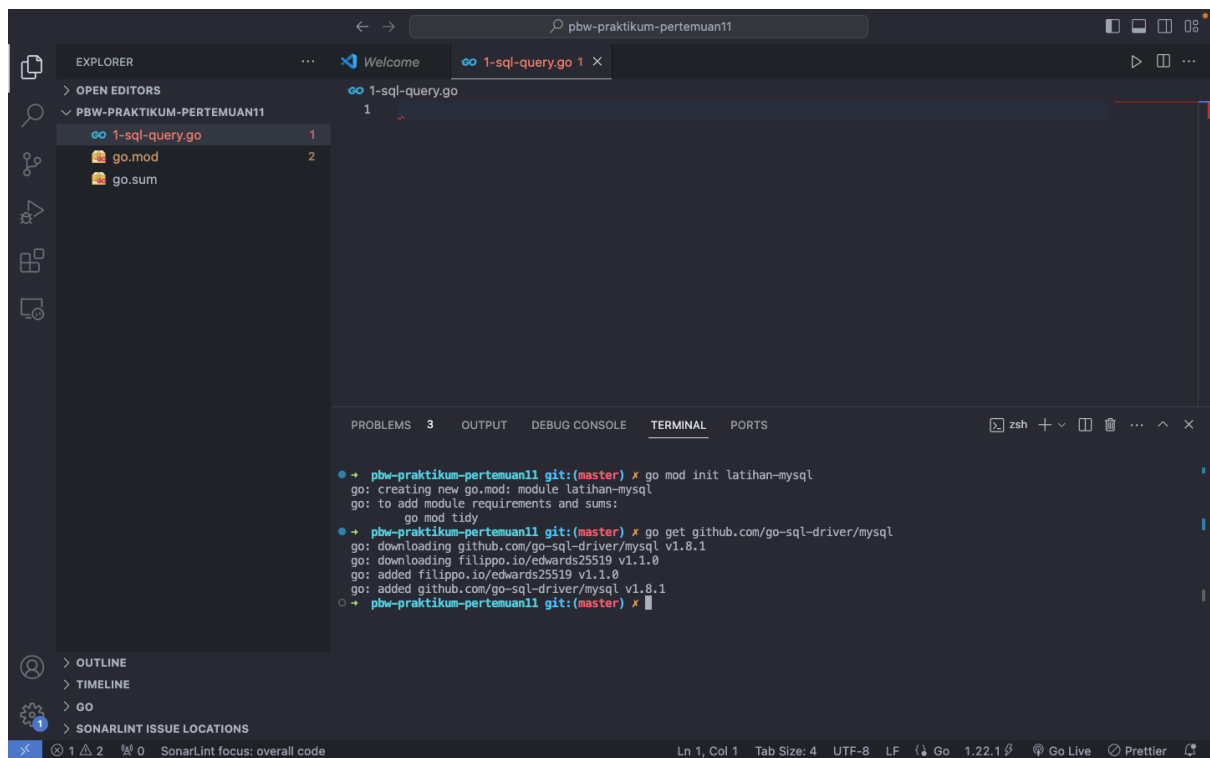
Nama : Siti Ghumaisa
NPM : 4522210131
Kelas : A
Semester : 4 (Genap)

Dosen :

ADI WAHYU PRIBADI ,S.SI.,M.KOM
S1-Teknik Informatika
Fakultas Teknik Universitas Pancasila
2023/2024

Link git-hub <https://github.com/Ghumaisa/PBW>

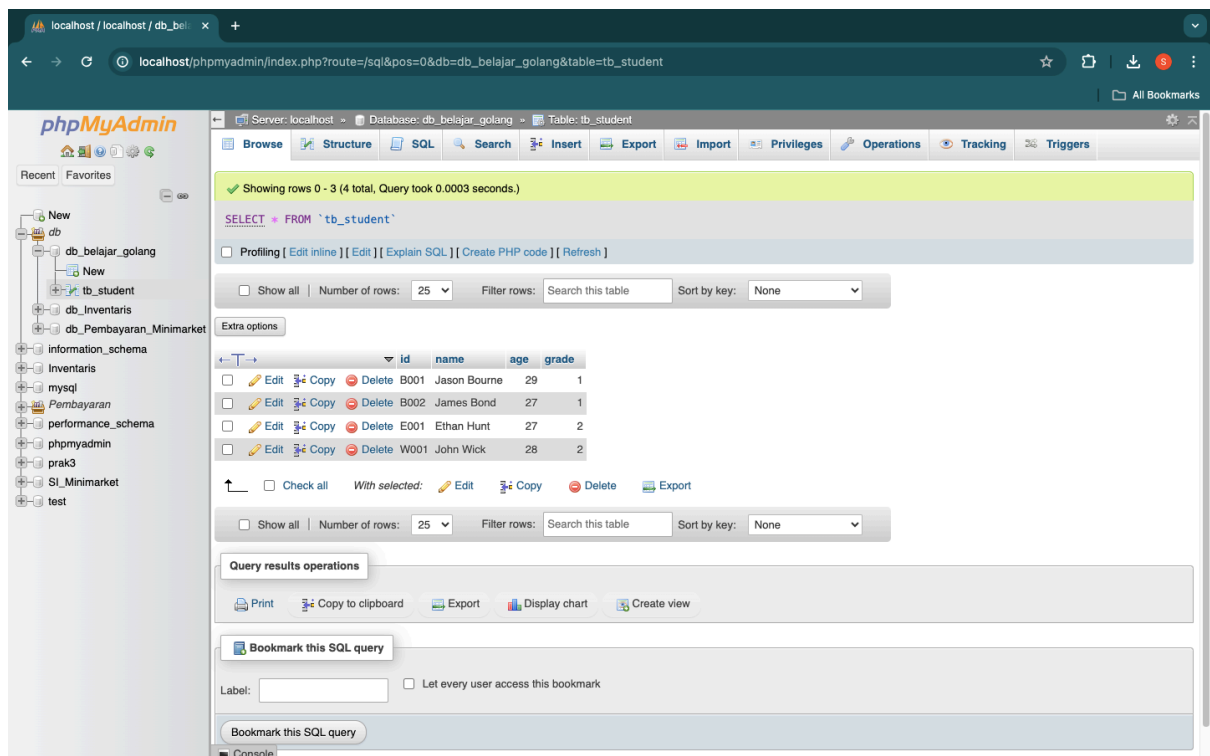
56.1. Instalasi Driver



The screenshot shows the Visual Studio Code interface with a Go project named 'pbw-praktikum-pertemuan11'. The Explorer panel on the left shows the project structure with files '1-sql-query.go', 'go.mod', and 'go.sum'. The Terminal panel at the bottom shows the following commands and output:

```
pbw-praktikum-pertemuan11 git:(master) x go mod init latihan-mysql
go: creating new go.mod: module latihan-mysql
go: to add module requirements and sums:
go mod tidy
pbw-praktikum-pertemuan11 git:(master) x go get github.com/go-sql-driver/mysql
go: downloading github.com/go-sql-driver/mysql v1.8.1
go: downloading filippo.io/edwards25519 v1.1.0
go: added filippo.io/edwards25519 v1.1.0
go: added github.com/go-sql-driver/mysql v1.8.1
```

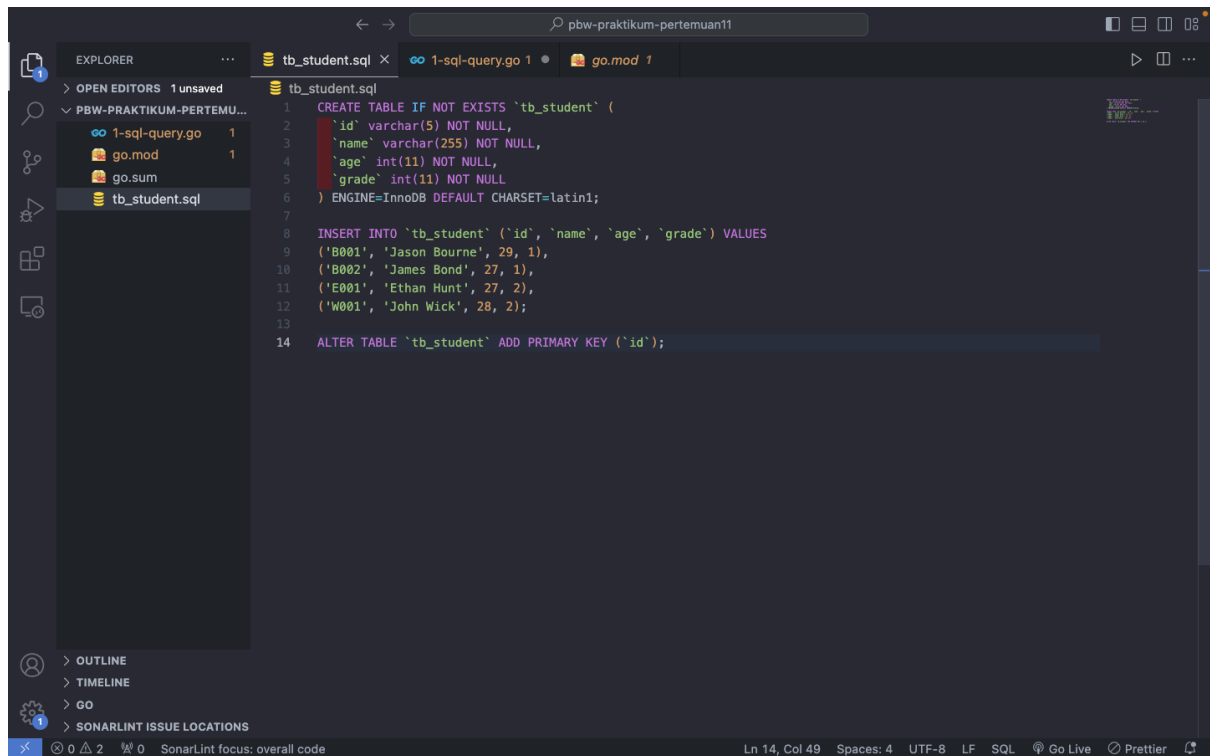
56.2. Setup Database



The screenshot shows the phpMyAdmin interface for a local database named 'db_belajar_golang'. The 'tb_student' table is selected, and the 'Structure' tab is active. The table contains 3 rows of data:

id	name	age	grade
B001	Jason Bourne	29	1
B002	James Bond	27	1
E001	Ethan Hunt	27	2

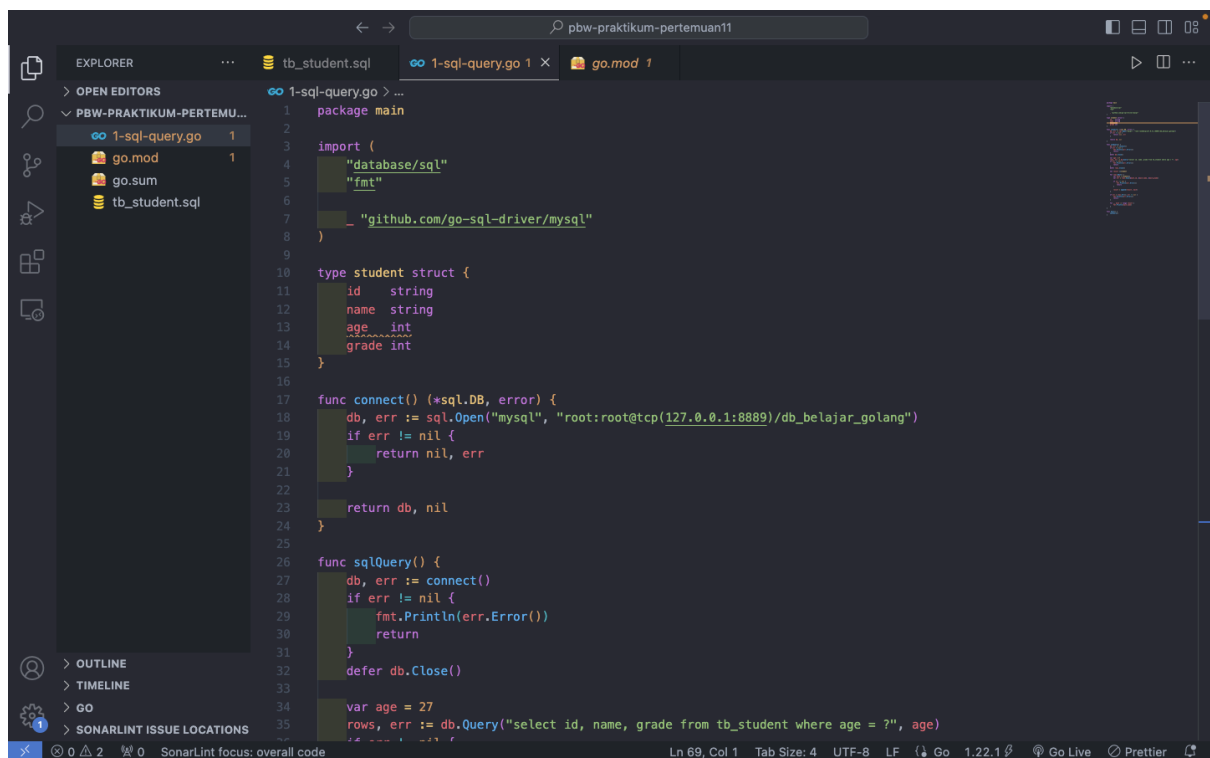
The interface also shows the 'Query results operations' section with options to print, copy to clipboard, export, display chart, and create view. The 'Bookmark this SQL query' section is also visible.



```
1 CREATE TABLE IF NOT EXISTS `tb_student` (  
2   `id` varchar(5) NOT NULL,  
3   `name` varchar(255) NOT NULL,  
4   `age` int(11) NOT NULL,  
5   `grade` int(11) NOT NULL  
6 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;  
7  
8 INSERT INTO `tb_student` (`id`, `name`, `age`, `grade`) VALUES  
9   ('B001', 'Jason Bourne', 29, 1),  
10  ('B002', 'James Bond', 27, 1),  
11  ('E001', 'Ethan Hunt', 27, 2),  
12  ('W001', 'John Wick', 28, 2);  
13  
14 ALTER TABLE `tb_student` ADD PRIMARY KEY (`id`);
```

56.3. Membaca Data Dari MySQL Server

Screenshoot Source Code



```
1 package main  
2  
3 import (  
4   "database/sql"  
5   "fmt"  
6  
7   _ "github.com/go-sql-driver/mysql"  
8 )  
9  
10 type student struct {  
11   id   string  
12   name string  
13   age  int  
14   grade int  
15 }  
16  
17 func connect() (*sql.DB, error) {  
18   db, err := sql.Open("mysql", "root:root@tcp(127.0.0.1:8889)/db_belajar_golang")  
19   if err != nil {  
20     return nil, err  
21   }  
22  
23   return db, nil  
24 }  
25  
26 func sqlQuery() {  
27   db, err := connect()  
28   if err != nil {  
29     fmt.Println(err.Error())  
30     return  
31   }  
32   defer db.Close()  
33  
34   var age = 27  
35   rows, err := db.Query("select id, name, grade from tb_student where age = ?", age)
```

This screenshot shows the implementation of the `sqlQuery` function in `1-sql-query.go`. The function takes an `age` parameter and queries the `tb_student` table for students of that age. It uses `db.Query` to execute the query, then iterates through the results using `rows.Next` and `rows.Scan` to populate a slice of `student` structs. The function also includes error handling for query execution and row scanning. The `main` function calls `sqlQuery` and prints the names of the returned students.

```
26 func sqlQuery() {
27     var age = 22
28     rows, err := db.Query("select id, name, grade from tb_student where age = ?", age)
29     if err != nil {
30         fmt.Println(err.Error())
31         return
32     }
33     defer rows.Close()
34
35     var result []student
36
37     for rows.Next() {
38         var each = student{}
39         var err = rows.Scan(&each.id, &each.name, &each.grade)
40
41         if err != nil {
42             fmt.Println(err.Error())
43             return
44         }
45
46         result = append(result, each)
47     }
48
49     if err = rows.Err(); err != nil {
50         fmt.Println(err.Error())
51         return
52     }
53
54     for _, each := range result {
55         fmt.Println(each.name)
56     }
57 }
58
59 func main() {
60     sqlQuery()
61 }
```

Screenshoot Output

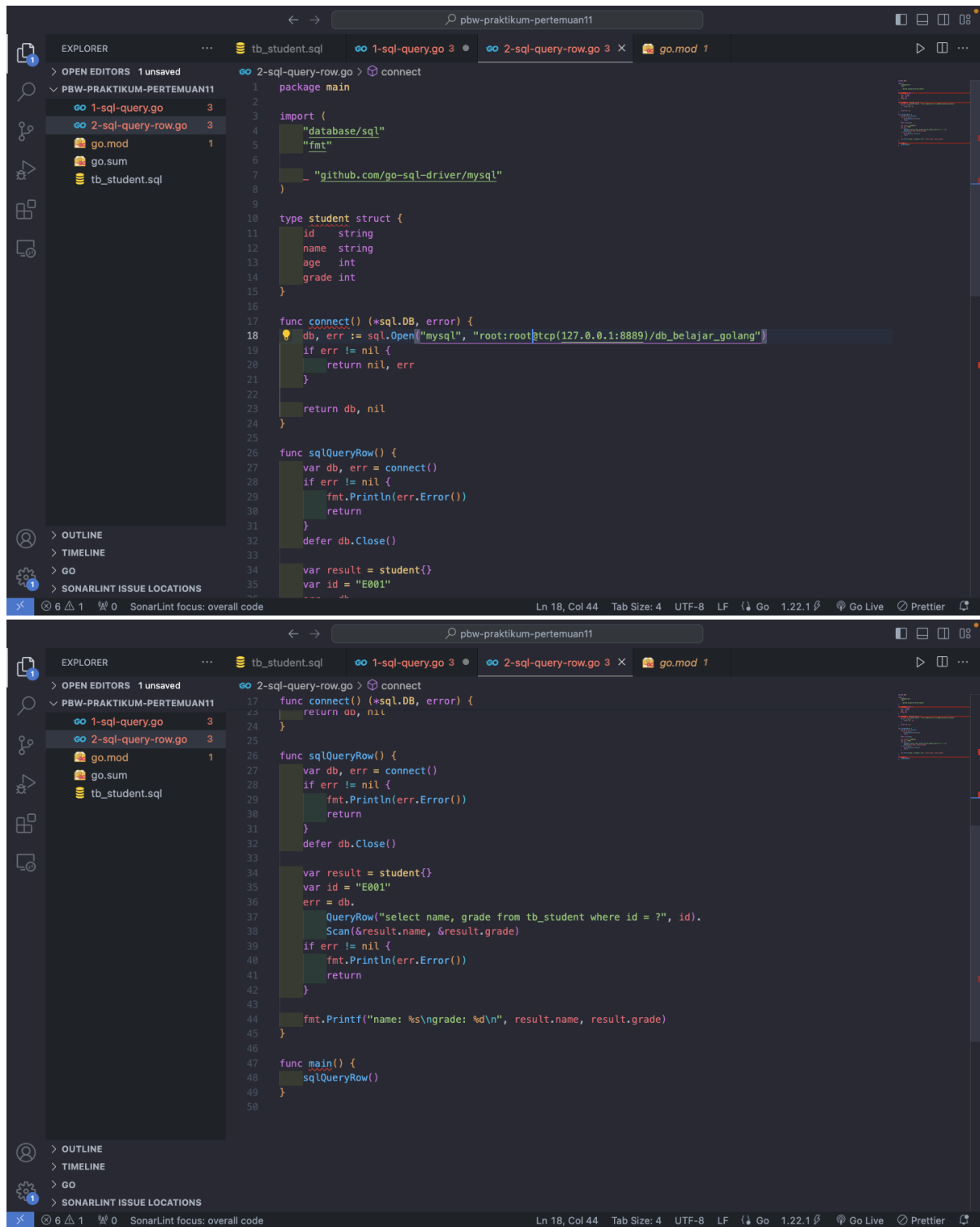
This screenshot shows the implementation of the `connect` function in `1-sql-query.go`, which establishes a connection to a MySQL database. The `student` struct is defined with fields `id`, `name`, `age`, and `grade`. The `connect` function uses `sql.Open` to create a new database connection. The `main` function calls `connect` and then `sqlQuery`. The terminal output shows the execution of the program, displaying the names of the students: James Bond and Ethan Hunt.

```
8
9
10 type student struct {
11     id    string
12     name  string
13     age   int
14     grade int
15 }
16
17 func connect() (*sql.DB, error) {
18     db, err := sql.Open("mysql", "root:root@tcp(127.0.0.1:8889)/db_belajar_golang")
19     if err != nil {
20         return nil, err
21     }
22
23     return db, nil
24 }
25
26 func sqlQuery() {
27     db, err := connect()
28     if err != nil {
29         return
30     }
31
32     sqlQuery()
33 }
```

Terminal Output:

```
pbw-praktikum-pertemuan11 git:(master) x go run 1-sql-query.go
James Bond
Ethan Hunt
pbw-praktikum-pertemuan11 git:(master) x
```

56.4. Membaca 1 Record Data Menggunakan Method Screenshoot Source Code



The image displays two screenshots of a Visual Studio Code editor window, showing Go source code for a database application. The editor is configured with a dark theme and the Go extension is installed.

Top Screenshot: The Explorer pane shows the project structure with files like `1-sql-query.go`, `2-sql-query-row.go`, `go.mod`, `go.sum`, and `tb_student.sql`. The main editor displays the code for `2-sql-query-row.go`, which includes a `connect` function and a `sqlQueryRow` function. The `connect` function establishes a connection to a MySQL database using the `github.com/go-sql-driver/mysql` driver. The `sqlQueryRow` function defines a `student` struct and attempts to connect to the database.

```
1 package main
2
3 import (
4     "database/sql"
5     "fmt"
6
7     _ "github.com/go-sql-driver/mysql"
8 )
9
10 type student struct {
11     id    string
12     name  string
13     age   int
14     grade int
15 }
16
17 func connect() (*sql.DB, error) {
18     db, err := sql.Open("mysql", "root:root@tcp(127.0.0.1:8889)/db_belajar_golang")
19     if err != nil {
20         return nil, err
21     }
22     return db, nil
23 }
24
25 func sqlQueryRow() {
26     var db, err = connect()
27     if err != nil {
28         fmt.Println(err.Error())
29         return
30     }
31     defer db.Close()
32
33     var result = student{}
34     var id = "E001"
```

Bottom Screenshot: This screenshot shows the continuation of the `sqlQueryRow` function and the `main` function. The `sqlQueryRow` function uses `QueryRow` to execute a SQL query to retrieve a single record from the `tb_student` table. The `main` function calls `sqlQueryRow`.

```
17 func connect() (*sql.DB, error) {
18     return db, nil
19 }
20
21 func sqlQueryRow() {
22     var db, err = connect()
23     if err != nil {
24         fmt.Println(err.Error())
25         return
26     }
27     defer db.Close()
28
29     var result = student{}
30     var id = "E001"
31     err = db.
32         QueryRow("select name, grade from tb_student where id = ?", id).
33         Scan(&result.name, &result.grade)
34     if err != nil {
35         fmt.Println(err.Error())
36         return
37     }
38     fmt.Printf("name: %s\ngrade: %d\n", result.name, result.grade)
39 }
40
41 func main() {
42     sqlQueryRow()
43 }
```

Screenshoot Output

The screenshot shows a VS Code editor with the following components:

- EXPLORER:** Shows a project named 'PBW-PRAKTIKUM-PERTEMUAN11' with files: `1-sql-query.go`, `2-sql-query-row.go`, `go.mod`, `go.sum`, and `tb_student.sql`.
- EDITOR:** Displays the code for `2-sql-query-row.go`. The code includes a `connect` function and a `sqlQueryRow` function that queries a database for a student's name and grade.
- TERMINAL:** Shows the output of running `go run 2-sql-query-row.go`. The output is:

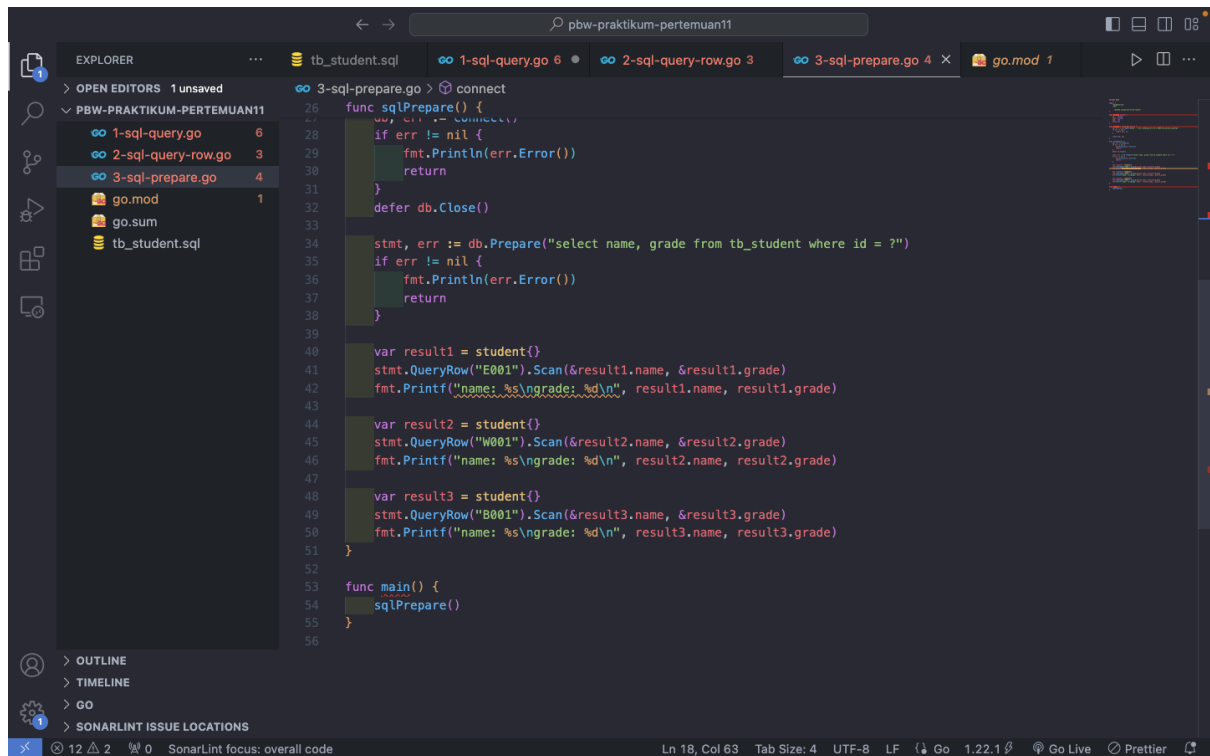
```
name: Ethan Hunt
grade: 2
```

56.5. Eksekusi Query Menggunakan Prepare()

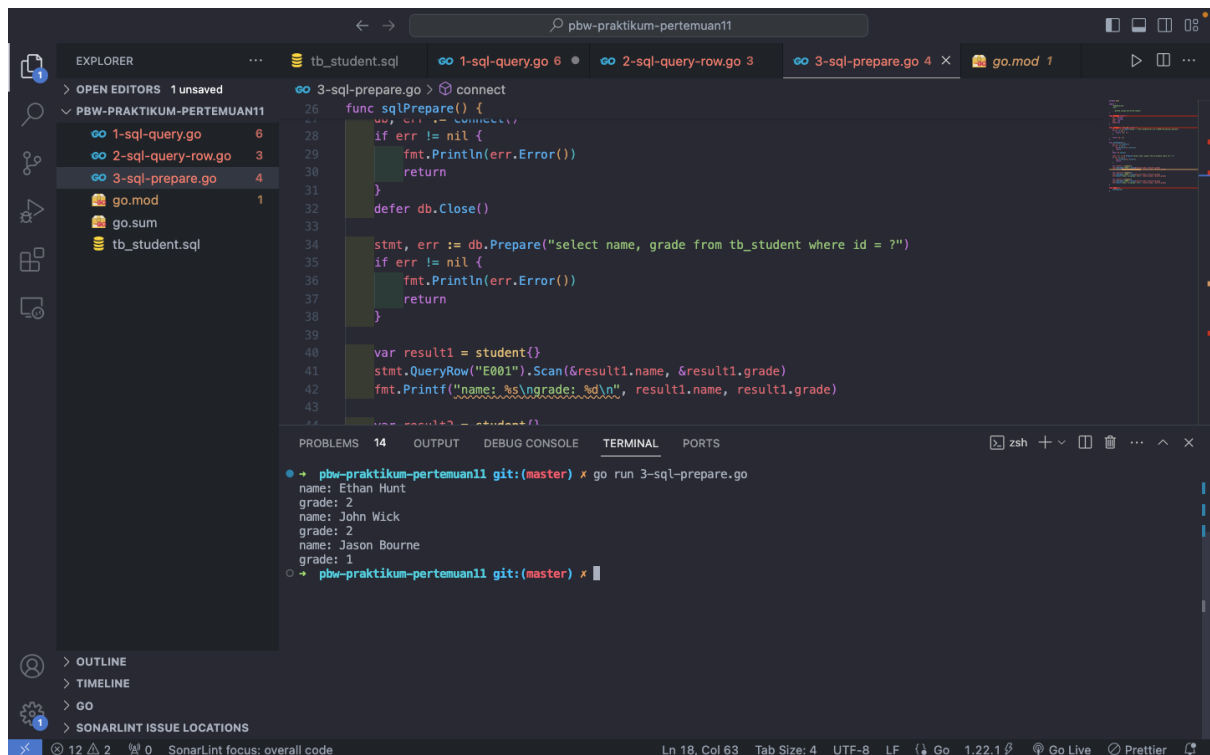
Screenshoot Source Code

The screenshot shows a VS Code editor with the following components:

- EXPLORER:** Shows a project named 'PBW-PRAKTIKUM-PERTEMUAN11' with files: `1-sql-query.go`, `2-sql-query-row.go`, `3-sql-prepare.go`, `go.mod`, `go.sum`, and `tb_student.sql`.
- EDITOR:** Displays the code for `3-sql-prepare.go`. The code includes a `connect` function, a `sqlPrepare` function, and a `stmt` variable that uses `db.Prepare` to execute a query.



Screenshoot Output



56.6. Insert, Update, & Delete Data Menggunakan Exec()

Screenshoot Source Code

```
1 package main
2
3 import (
4     "database/sql"
5     "fmt"
6     _ "github.com/go-sql-driver/mysql"
7 )
8
9 type student struct {
10     id    string
11     name  string
12     age   int
13     grade int
14 }
15
16
17 func connect() (*sql.DB, error) {
18     db, err := sql.Open("mysql", "root:root@tcp(127.0.0.1:8889)/db_belajar_golang")
19     if err != nil {
20         return nil, err
21     }
22     return db, nil
23 }
24
25
26 func sqlExec() {
27     db, err := connect()
28     if err != nil {
29         fmt.Println(err.Error())
30         return
31     }
32     defer db.Close()
33
34     _, err = db.Exec("insert into tb_student values (?, ?, ?, ?)", "G001", "Galahad", 29, 2)
35     if err != nil {
36         fmt.Println(err.Error())
37     }
38     fmt.Println("insert success!")
39
40     _, err = db.Exec("update tb_student set age = ? where id = ?", 28, "G001")
41     if err != nil {
42         fmt.Println(err.Error())
43         return
44     }
45     fmt.Println("update success!")
46
47     _, err = db.Exec("delete from tb_student where id = ?", "G001")
48     if err != nil {
49         fmt.Println(err.Error())
50         return
51     }
52     fmt.Println("delete success!")
53 }
54
55
56 func main() {
57     sqlExec()
58 }
59
```


Screenshoot Output

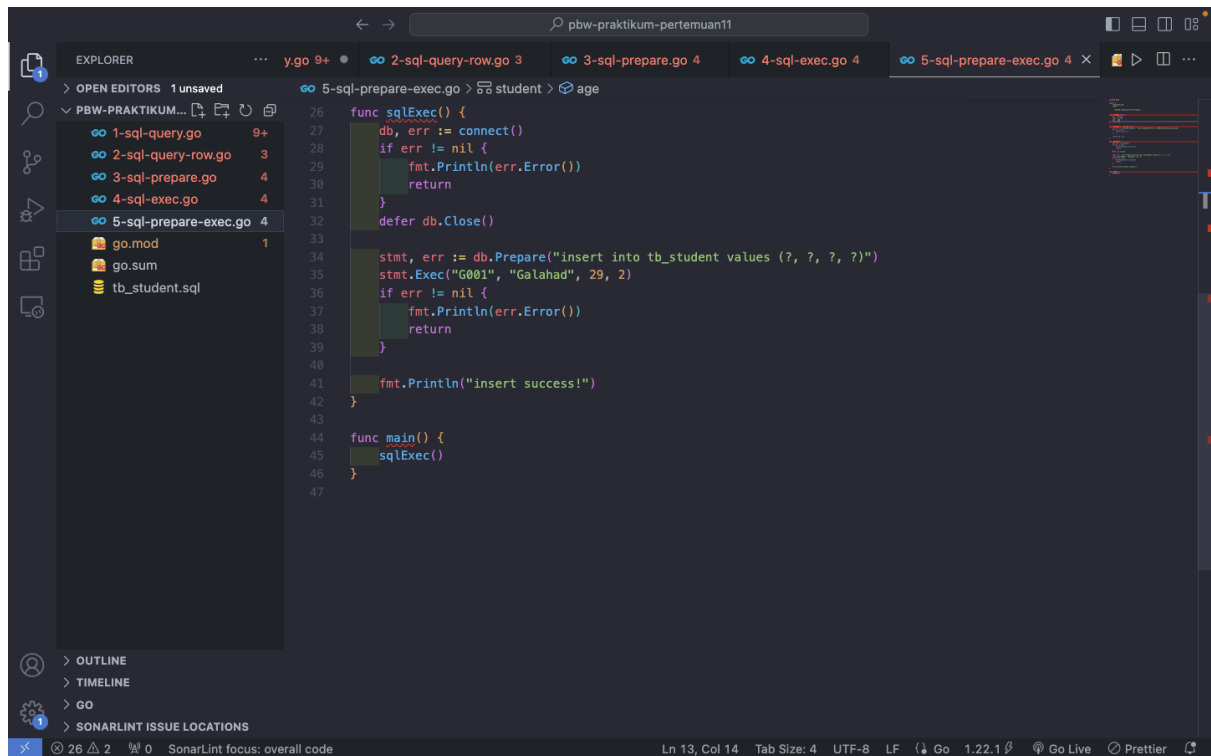
```
func sqlExec() {  
    _, err = db.Exec("insert into tb_student values (?, ?, ?, ?)", "G001", "Galahad", 29, 2)  
    if err != nil {  
        fmt.Println(err.Error())  
        return  
    }  
    fmt.Println("insert success!")  
  
    _, err = db.Exec("update tb_student set age = ? where id = ?", 28, "G001")  
    if err != nil {  
        fmt.Println(err.Error())  
        return  
    }  
    fmt.Println("update success!")  
  
    _, err = db.Exec("delete from tb_student where id = ?", "G001")  
    if err != nil {  
        fmt.Println(err.Error())  
        return  
    }  
    fmt.Println("delete success!")  
}
```

```
pbw-praktikum-pertemuan11 git:(master) x go run 4-sql-exec.go  
insert success!  
update success!  
delete success!  
pbw-praktikum-pertemuan11 git:(master) x
```

5-sql-prepare-exec.go

Screenshoot Source Code

```
package main  
  
import (  
    "database/sql"  
    "fmt"  
    _ "github.com/go-sql-driver/mysql"  
)  
  
type student struct {  
    id      string  
    name    string  
    age     int  
    grade   int  
}  
  
func connect() (*sql.DB, error) {  
    db, err := sql.Open("mysql", "root:root@tcp(127.0.0.1:8889)/db_belajar_golang")  
    if err != nil {  
        return nil, err  
    }  
    return db, nil  
}  
  
func sqlExec() {  
    db, err := connect()  
    if err != nil {  
        fmt.Println(err.Error())  
        return  
    }  
    defer db.Close()  
  
    stmt, err := db.Prepare("insert into tb_student values (?, ?, ?, ?)")  
    if err != nil {  
        fmt.Println(err.Error())  
        return  
    }  
    stmt.Exec("G001", "Galahad", 29, 2)  
}
```



Screenshoot Output

