Java

```java
class SQL {

    public SQL(List<String> names, List<Integer> columns) {

    }

    public boolean ins(String name, List<String> row) {

    }

    public void rmv(String name, int rowId) {

    }

    public String sel(String name, int rowId, int columnId) {

    }

    public List<String> exp(String name) {

    }
}

/**
 * Your SQL object will be instantiated and called as such:
 * SQL obj = new SQL(names, columns);
 * boolean param_1 = obj.ins(name,row);
 * obj.rmv(name,rowId);
 * String param_3 = obj.sel(name,rowId,columnId);
 * List<String> param_4 = obj.exp(name);
```

```
 */
--------------------------------------------------------------------------
```

JavaScript

```
/**
 * @param {string[]} names
 * @param {number[]} columns
 */
var SQL = function(names, columns) {

};

/**
 * @param {string} name
 * @param {string[]} row
 * @return {boolean}
 */
SQL.prototype.ins = function(name, row) {

};

/**
 * @param {string} name
 * @param {number} rowId
 * @return {void}
 */
SQL.prototype.rmv = function(name, rowId) {

};

/**
```

```
 * @param {string} name
 * @param {number} rowId
 * @param {number} columnId
 * @return {string}
 */
SQL.prototype.sel = function(name, rowId, columnId) {

};

/**
 * @param {string} name
 * @return {string[]}
 */
SQL.prototype.exp = function(name) {

};

/**
 * Your SQL object will be instantiated and called as such:
 * var obj = new SQL(names, columns)
 * var param_1 = obj.ins(name,row)
 * obj.rmv(name,rowId)
 * var param_3 = obj.sel(name,rowId,columnId)
 * var param_4 = obj.exp(name)
 */
```

TypeScript

```
class SQL {
    constructor(names: string[], columns: number[]) {
```

```
    }

    ins(name: string, row: string[]): boolean {

    }

    rmv(name: string, rowId: number): void {

    }

    sel(name: string, rowId: number, columnId: number): string {

    }

    exp(name: string): string[] {

    }
}

/**
 * Your SQL object will be instantiated and called as such:
 * var obj = new SQL(names, columns)
 * var param_1 = obj.ins(name,row)
 * obj.rmv(name,rowId)
 * var param_3 = obj.sel(name,rowId,columnId)
 * var param_4 = obj.exp(name)
 */
```

C++

```
class SQL {
```

```
public:
    SQL(vector<string>& names, vector<int>& columns) {

    }

    bool ins(string name, vector<string> row) {

    }

    void rmv(string name, int rowId) {

    }

    string sel(string name, int rowId, int columnId) {

    }

    vector<string> exp(string name) {

    }
};

/**
 * Your SQL object will be instantiated and called as such:
 * SQL* obj = new SQL(names, columns);
 * bool param_1 = obj->ins(name,row);
 * obj->rmv(name,rowId);
 * string param_3 = obj->sel(name,rowId,columnId);
 * vector<string> param_4 = obj->exp(name);
 */
```

C#

```
public class SQL {

    public SQL(IList<string> names, IList<int> columns) {

    }

    public bool Ins(string name, IList<string> row) {

    }

    public void Rmv(string name, int rowId) {

    }

    public string Sel(string name, int rowId, int columnId) {

    }

    public IList<string> Exp(string name) {

    }
}

/**
 * Your SQL object will be instantiated and called as such:
 * SQL obj = new SQL(names, columns);
 * bool param_1 = obj.Ins(name,row);
 * obj.Rmv(name,rowId);
 * string param_3 = obj.Sel(name,rowId,columnId);
 * IList<string> param_4 = obj.Exp(name);
```

```
 */
```

Kotlin

```kotlin
class SQL(names: List<String>, columns: List<Int>) {

    fun ins(name: String, row: List<String>): Boolean {

    }

    fun rmv(name: String, rowId: Int) {

    }

    fun sel(name: String, rowId: Int, columnId: Int): String {

    }

    fun exp(name: String): List<String> {

    }

}

/**
 * Your SQL object will be instantiated and called as such:
 * var obj = SQL(names, columns)
 * var param_1 = obj.ins(name,row)
 * obj.rmv(name,rowId)
 * var param_3 = obj.sel(name,rowId,columnId)
 * var param_4 = obj.exp(name)
```

```
 */
```

Go

```go
type SQL struct {

}

func Constructor(names []string, columns []int) SQL {

}

func (this *SQL) Ins(name string, row []string) bool {

}

func (this *SQL) Rmv(name string, rowId int)  {

}

func (this *SQL) Sel(name string, rowId int, columnId int) string {

}

func (this *SQL) Exp(name string) []string {

}
```

```
/**
 * Your SQL object will be instantiated and called as such:
 * obj := Constructor(names, columns);
 * param_1 := obj.Ins(name,row);
 * obj.Rmv(name,rowId);
 * param_3 := obj.Sel(name,rowId,columnId);
 * param_4 := obj.Exp(name);
 */
----------------------------------------------------------------------
```