Q1.1：

```
SELECT
tc.constraint_name,
tc.table_name,
tc.constraint_type,
pg_get_constraintdef(pc.oid) AS constraint_definition
FROM
information_schema.table_constraints AS tc
JOIN
pg_constraint AS pc ON tc.constraint_name = pc.conname
WHERE
tc.table_name IN ('staff', 'dept', 'sales', 'customer')
AND tc.constraint_schema = 'public';
```

```
 constraint_name | table_name | constraint_type |  constraint_definition
-----------------+------------+-----------------+-------------------------
 pk_staffno      | staff      | PRIMARY KEY     | PRIMARY KEY (staffno)
 pk_deptno       | dept       | PRIMARY KEY     | PRIMARY KEY (deptno)
 pk_customerno   | customer   | PRIMARY KEY     | PRIMARY KEY (customerno)
(3 rows)
```

Q1.2:

```
ALTER TABLE SALES ADD CONSTRAINT PK_SALENO PRIMARY KEY (SaleNo);

ALTER TABLE DEPT ADD CONSTRAINT UN_DNAME UNIQUE(DName);

ALTER TABLE STAFF ADD CONSTRAINT CK_STAFFNAME CHECK (StaffName IS NOT NULL);

ALTER TABLE DEPT ADD CONSTRAINT CK_DNAME CHECK (DName IS NOT NULL);

ALTER TABLE CUSTOMER ADD CONSTRAINT CK_CNAME CHECK (CName IS NOT NULL);

ALTER TABLE SALES ADD CONSTRAINT CK_RECEIPTNO CHECK (ReceiptNo IS NOT NULL);

ALTER TABLE SALES ADD CONSTRAINT CK_AMOUNT CHECK (Amount>0);

ALTER TABLE STAFF ADD CONSTRAINT CK_POSITION
CHECK (Position IN ('Group Manager','Group Assistant', 'Group Member','Team Leader', 'Branch Manager'));

ALTER TABLE SALES ADD CONSTRAINT CK_SERCIVETYPE
CHECK (ServiceType IN ('Software Installation','Software Repair', 'Training','Consultation','Data Recovery'));

ALTER TABLE SALES ADD CONSTRAINT CK_PAYMENTTYPE
```

```
CHECK (PaymentType IN ('Debit', 'Cash', 'Credit'));

ALTER TABLE SALES ADD CONSTRAINT CK_GST
CHECK (GST IN ('Yes','No'));

ALTER TABLE STAFF ADD CONSTRAINT FK_DEPTNO
FOREIGN KEY (DeptNo) REFERENCES DEPT(DeptNo);

ALTER TABLE SALES ADD CONSTRAINT FK_STAFFNO
FOREIGN KEY (ServedBy) REFERENCES STAFF(StaffNo);

ALTER TABLE SALES ADD CONSTRAINT FK_CUSTOMERNO
FOREIGN KEY (CustomerNo) REFERENCES CUSTOMER(CustomerNo);
```

```
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
```

Q2.1

```
CREATE SEQUENCE PNO_SEQ
MINVALUE 10000
INCREMENT BY 1
START WITH 10000;
```

```
assignment1=# CREATE SEQUENCE PNO_SEQ
MINVALUE 10000
INCREMENT BY 1
START WITH 10000;
CREATE SEQUENCE
```

Q2.2

```
CREATE OR REPLACE FUNCTION
UDF_BI_PNO()
RETURNS TRIGGER AS $$
BEGIN
IF NEW.SaleNo IS NULL THEN
NEW.SaleNo := nextval('PNO_SEQ');
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER BI_PNO
BEFORE INSERT ON SALES
FOR EACH ROW
EXECUTE FUNCTION
UDF_BI_PNO();
```

```
CREATE FUNCTION
CREATE TRIGGER
```

Q2.3

```
CREATE OR REPLACE FUNCTION UDF_TOP_DISCOUNT()
RETURNS TRIGGER AS $$
BEGIN
IF NEW.customerNo IN (
SELECT customerNo FROM SALES
GROUP BY CustomerNo
    HAVING SUM(amount)=(
        SELECT MAX(Total) FROM
        (
        SELECT SUM(amount) AS Total FROM SALES
```

```
            GROUP BY CustomerNo) total
        )
)
THEN NEW.amount := NEW.amount * 0.85;
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER TOP_DISCOUNT
BEFORE INSERT ON SALES
FOR EACH ROW
EXECUTE FUNCTION UDF_TOP_DISCOUNT();
```

```
CREATE FUNCTION
CREATE TRIGGER
```

Q2.4

```
CREATE OR REPLACE FUNCTION UDF_SUNSHINE_DEPT()
RETURNS TRIGGER AS $$
BEGIN
IF NEW.servedby IN (
SELECT s.staffno FROM STAFF s
JOIN DEPT d ON d.deptno = s.deptno
WHERE d.dname = 'SALES - Sunshine'
)
THEN NEW.paymenttype :='Cash';

IF NEW.servicetype='Data Recovery' THEN
NEW.amount:=NEW.amount*0.7;
END IF;
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER SUNSHINE_DEPT
BEFORE INSERT ON SALES
FOR EACH ROW
EXECUTE FUNCTION UDF_SUNSHINE_DEPT();
```

```
CREATE FUNCTION
CREATE TRIGGER
```

Q2.5

```
CREATE OR REPLACE FUNCTION UDF_TIME_CHECK()
RETURNS TRIGGER AS $$
BEGIN
IF NEW.customerno IN(
SELECT s.customerno FROM SALES s
JOIN STAFF st ON st.staffno = s.servedby
JOIN DEPT d ON d.deptno=st.deptno
WHERE d.dlocation NOT IN(
SELECT d2.dlocation FROM STAFF st2
JOIN DEPT d2 ON d2.deptno = st2.deptno
WHERE st2.staffno = NEW.servedby)
AND s.saletime >= NEW.saletime - INTERVAL '5 minutes'
AND s.saletime<NEW.saletime)
THEN RAISE EXCEPTION 'Invalid';
END IF;

IF (CAST(NEW.saletime AS time) < TIME '06:00' OR CAST(NEW.saletime AS time) > TIME '21:00')
AND NEW.servedby NOT IN (
SELECT st.staffno FROM STAFF st
WHERE st.position ILIKE '%manager%')
THEN RAISE EXCEPTION 'Invalid';
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER TIME_CHECK
BEFORE INSERT ON sales
FOR EACH ROW
EXECUTE FUNCTION UDF_TIME_CHECK();
```

```
CREATE FUNCTION
CREATE TRIGGER
```